INSTITUTE FOR DEFENSE ANALYSES

An Initial Look at Alternative Computing Technologies for the Intelligence Community

Lance Joneckis - IDA
David Koester - MITRE Corporation
Joshua Alspector - IDA

January 2014

Approved for public release;
distribution is unlimited.

IDA Paper P-5114
Log: H 14-000128

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

About This Publication

This work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-14-D-0001, Project ET-2-2954.22, "Alternative Computational Technology." This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Army Contracting Command contract number W91WAW-12-C-0017. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposed notwithstanding any copyright annotation thereon.

INSTITUTE FOR DEFENSE ANALYSES

IDA Paper P-5114

An Initial Look at Alternative Computing Technologies for the Intelligence Community

Lance Joneckis - IDA
David Koester - MITRE Corporation
Joshua Alspector - IDA

Contents

1 Executive Summary

We have broadly surveyed the computing landscape as it relates to the problems of interest to the intelligence community (IC). We did not find a single alternative computing technology (ACT) that is universally applicable to the wide range of IC applications, although particular solutions are suited to individual application classes. We therefore recommend that the IC consider an application-driven, holistic design approach that spans a broad range of technologies and computational models. The application classes we considered were discrete math, big data, distributed sensing and processing, scientific/numerical simulation, and robotics/autonomous systems. Holistic design could extend beyond a single technology to include integration of multiple, heterogeneous ACTs as needed to meet application/architecture requirements.

To meet mission requirements, the IC has been a leader in the development and deployment of computing and computing technologies. The computing design space in which the IC has worked runs the gamut from employing commodity components and systems to advanced special-purpose devices (SPDs). For those missions where commodity components have limited utility, the IC must question the implicit assumptions underlying current computing system design choices. This questioning must extend deeply, even to the fundamental model of computation used. Instead of asking which complementary metal oxide semiconductor (CMOS) chip we should use for a problem, we should ask about the chip we should design. Instead of what digital means of computation should we use, we might ask if we should even be solving this problem digitally. Clearly, a practice of routinely returning to fundamental questions bristles against practical concerns (such as schedule and cost), but it is the approach most likely to meet some of the unique, future needs of the IC.

The survey of the computing landscape documented in this paper extends far beyond the traditional classical digital style of computation and conventional digital CMOS technology. It even extends beyond quantum computing. By *style* or *model* of computation, we refer to how the solution to a given problem is obtained on a given computing system. The classical digital model of computation is embodied in what we might call *conventional* CMOS computing, ranging from the processing done in smart phones to the number-crunching done in high-performance computing (HPC) clusters. In the classical digital model data are represented digitally, and solutions to problems are computed via algorithms in a series of discrete steps. By contrast, in analog computing, data are represented by continuous values, and the solution to a problem is *computed* through the evolution of a dynamical system.

In the following paragraph, we describe the four fundamental styles of computing we considered: the classical digital, neuro-inspired, analog, and quantum digital. For each style of computing, we also discuss the technologies that may be used to realize a computing platform that solves problems in that style. We distinguish between technologies and platforms. For example, CMOS is a technology, while *digital CMOS*, with its associated infrastructure (including, for example, design and fabrication tools, and related technologies like dynamic random-access memory (DRAM)), is a device platform for realizing classical digital computing. In general, a device platform comprises the state variable (the means used to represent the computational unit), the material (the stuff that hosts the state variable), and the device (the means to manipulate state).

Classical Digital Computing: Classical digital computing, in which bits represent information and computation corresponds to sequences of elementary logic operations, dominates computing today. Conventional digital computing has, indeed, dominated the computer engineering and computer science communities for decades. We considered three platforms for conventional digital computing: CMOS and Beyond CMOS, superconducting, and biomolecular.

The CMOS and Beyond CMOS platform has been, and will continue to be, the dominant platform even after the CMOS transistor has reached its practical scaling limit, around 2024. Today, the number of transistors manufactured on a single CMOS 300-mm diameter wafer can approach one trillion. Other than biological processes, no other manufacturing process can come close to that low cost of compute element production. Industry will support the advancement of CMOS at the device (e.g., transistor) level throughout that time frame, with little need for United States Government (USG) investment. USG investment at the systems level (chip design through algorithm), however, can leverage industry's massive underlying investment and thus may have considerable impact. The two prime areas for USG investment in the CMOS and Beyond CMOS platform are holistic design and the power costs of high-bandwidth, low-latency communication. Most of the energy dissipation in CMOS is entailed in moving bits, and holistic designs that innovate across the spectrum of hardware to algorithm will be needed to obtain more efficient use of communication resources. Considering the entire technology stack can drive solutions that are optimized over the whole.

The cryogenic superconducting platform offers the potential of increasing computational power efficiency by 10×–1,000×. CMOS and Josephson junction superconducting logic elements inherently draw comparable amounts of power, but the latter enjoys a substantial advantage in needing to expend almost no energy to move bits around. Applications that are communication intensive, which is the case for many of today's IC applications, will benefit the most from superconducting technology. USG investment will be required to move this platform forward through research to advanced development. Without eventual commercial adoption of superconducting technology, the USG will have to shoulder the burden of scaling the technology to increase performance, possibly a cost too great to bear alone.

The biomolecular platform offers the potential for power-efficient computing operating at the molecular level. While solutions to small instances of computationally hard problems have been demonstrated in this technology, extant biomolecular techniques have poor scalability and specificity. The molecular tile approach provides for more control over the calculation but suffers from errors unless the speed of the calculation proceeds slowly. For these reasons this platform has little value for computationally hard IC problems. The biomolecular research community is currently exploring simple logic operations using a number of diverse mechanisms *in vitro* and *in vivo*, targeted not at complex calculations but rather for control of natural and synthetic biological systems. Likely application areas for these results include medicine, energy production, and environmental engineering. We recommend no IC investment at present though other parts of the USG, notably National Institute of Health (NIH) and National Science Foundation (NSF) should consider investing. The IC should monitor this area for future applicability to problems of interest.

Neuro-Inspired Computing: Neuro-inspired computing is expected to have superior performance for problems involving discrimination and pattern recognition, for example, with applicability to processing big data. This style of computing attempts to emulate the computation in animals with the goal of providing similar capabilities in engineered systems. It is motivated by the recognition that there is no transistor-based classical digital architecture, either now or foreseen for the future, that can reproduce the capabilities of the human brain while dissipating only 15 W.

The neuro-inspired compute model is currently a work in progress because we have only a limited understanding of how the brain works. Today's neuro-inspired architectures and algorithms extrapolate generously from a minimal grounding in biology. Maturing these approaches will require better characterization of the brain's processes at both the biological and algorithmic levels. We recommend an emphasis on interdisciplinary research strongly rooted in biology and computer science and complemented by other relevant disciplines.

We considered two platforms for neuro-inspired computing: analog CMOS and analog biomolecular. The CMOS platform is amply suited for realizing neuron-like devices, but it is missing an efficient synapse to be used for distributed memory and coupling neurons. Biomolecular technology, particularly in the context of synthetic biology, is at present too immature to realize neuro-inspired computing, but does offer a potentially interesting approach for the long term. Although the drivers of synthetic biology are not computation, this area should be watched for a potentially disruptive impact on neuro-inspired computing.

Analog Computing: Analog computing represents information as real-valued quantities, with computation proceeding via a set of differential equations of motion. Analog computation has low precision, making it undesirable for computations requiring exact results. *Programming* an analog system consists of mapping the problem of interest onto a dynamical system; for example, it appears to be possible to solve Boolean equations—an application of interest to the IC—by mapping the problem onto a system whose solution is given by the solution of differential equations. That done, it is unclear whether the most efficient place to solve those differential equations is on an analog computer.

Analog computing is reasonably mature, and a variety of technologies, ranging from analog CMOS to the extreme of analog computation in cells, present feasible platforms. The "programming" and set-up of the analog computation can be challenging even when running studies of similar problems. The benefits of analog computing need, however, to be established before investing in extensions to the computational platform. We recommend a small research effort to explore mappings between IC hard problems and dynamical systems.

Quantum Digital Computing: Quantum digital computing has the potential to reduce computational complexity for select problems. Shor's algorithm gives a way to shrink the number of required operations (from exponential to polynomial) for integer factoring and is the main motivation behind the desire to build a quantum computer. Simply put, a quantum computer is more powerful than a classical computer when it comes to factoring because, in principle, it has to do less work. Shor's algorithm is a particular application of a more general capability of quantum computing—a limited solution of the hidden subgroup problem, and this application has far reaching implications.

Quantum computing has explored well over a dozen candidate material systems over the last 25 years. Trapped ions, superconducting qubits, and electron-spin in solid-state materials have survived and continue to progress. Despite this investment, major challenges persist. A *logical* qubit, (i.e., one whose integrity is guaranteed for the duration of a calculation) remains elusive. Also, practical systems will likely comprise millions of qubits, which will need to be able to co-exist on that scale in an architecture. A worthy intermediate goal, therefore, is a scalable logical qubit. Obtaining such a qubit will require approaching the problem not only from the perspective of a technology that makes the best single qubit, but also asking what technology best supports system requirements for the desired capability. In other words, the solution should be holistically designed for a specific application. Adopting such an approach begins to move us away from a technology-centric perspective and towards a platform-centric perspective for quantum computing.

Holistic design may work to focus research to useful classes of technologies. Meanwhile, no technology that is capable of scaling to large numbers of logical qubits and adequately stable to complete a calculation has been identified. Mission capabilities implemented by quantum digital computing must be weighed against the likely substantial research investment required.

Conclusion: As computing moves into the future, solutions will necessarily become more specialized to the problem at hand. The more the solution space is expanded, the greater the potential for high-performance solutions but also the more investment that will be required. No single approach that covers the range of computational problems of interest. Because computational capabilities result in operational capabilities for the IC, investing in multiple approaches or in one approach at the expense of others is fundamentally a strategic decision that has long-term implications. Any strategic investment decision needs to carefully consider operational needs and tradeoffs.

2 Findings and Recommendations

2.1 Global Recommendations

Finding: With power, space, and cooling as forcing functions, computational power efficiency will become ever more important. Application-specific solutions, on which the IC has historically relied for a leg up, will be even more important for the computationally hard problems of the future. Maximum efficiencies will come from the right choice of computational style and platform.

Recommendation (G-1): *Expand the design space to include a more diverse set of computational models and platforms.* The IC, to preserve its edge, should extend the concept of application specificity to encompass computational styles and platforms. Innovation should not be restricted to conventional classical digital computational models realized on limited CMOS platforms. Pairing a problem to the right combination of style and platform, for example, factoring integers on a quantum computer, or using neuro-inspired processing for pattern recognition can lead to enormous gains in efficiency over conventional approaches. Hybrid systems that mix more than one style, such as embedding analog computation for certain mathematical operations, may also provide a benefit.

The dominance of digital computation has displaced analog computation historically, and this dominance presents a barrier to fully establishing alternative computational styles, such as neuro-inspired computation and the supporting hardware. Progress is further compounded by the need for these alternative computational styles to exceed capabilities of a digital solution. This situation is appreciated and several efforts are under way that address these issue. Substantial progress will require a sustained long-term and strategic commitment to these alternative computational styles.

Finding: A potentially viable alternative technology to CMOS (and its projected commercial evolution) for general-purpose, classical digital computation is superconducting, which offers the potential for an estimated one to three orders of magnitude reduction in electrical power, though this reduction in power is far from being realized. The other ACT candidates that emerged from the study—specifically neuro-inspired, analog, and quantum—represent not new technologies, but different computation styles. Each could be implemented via a range of substrates. The relevance of quantum computing to the IC is undisputed, but the potential of the other computational styles to serve IC needs is not as well understood. Neuro-inspired computing should be efficient for discrimination and pattern matching with applicability to big data. Analog computing has the potential of efficiently solving a set of first-order differential equations. ACTs may offer significant potential; however, much yet needs to be learned, and that research will require investment.

Recommendation (G-2): *Algorithms are key.* Historically, the big leaps in our ability to factor numbers have come from new algorithms, with incremental assistance from progress in microelectronics. Improved algorithms have also have made important contributions in other areas, such as signal processing and machine learning. Like quantum computing, neuro-inspired and analog computing are attractive for their potential to stimulate the recasting of hard problems into forms that are more readily soluble. Also, just as efforts to realize quantum computing have required the joint contributions of mathematicians, computer scientists and physicists, the IC will need to mount interdisciplinary efforts to understand the implications of these other new computational models for its problem set.

Finding: The International Technology Roadmap for Semiconductors (ITRS) roadmap indicates the viability of CMOS for another 20 years. Industry is investing nearly $50 billion yearly to sustain its trajectory. The CMOS platform will continue to advance even after CMOS reaches its practical scaling limit.

Recommendation (G-3): *Know where to set the bar.* To determine whether a candidate merits serious consideration as an alternate computing technology, the IC should compare its likely performance with the projections for CMOS on the ITRS roadmap.

Finding: No single alternative to CMOS can meet the ICs full range of computational needs of the IC. Quantum computing, for example, is superior for factoring integers, but neuro-inspired computation is superior for pattern matching,

Recommendation (G-4): *Consider these preliminary findings.* This study has identified benefits and limitations of ACTs. Since substantial investment will be needed to bring any individual ACT to the level of practicability, a detailed study of critical capabilities and gaps should be done before selecting or eliminating an ACT from consideration.

2.2 Compute model-specific solutions

2.2.1 Classical Digital Computing

Finding: The ITRS roadmap has a comprehensive strategy for emerging research material and devices. It has also defined 5 difficult challenges in the areas of memory, CMOS, Beyond CMOS, and heterogenous integration to focus the research. It is not clear that USG investment in this area will make a difference. Though the USG should closely monitor this area, and may need to make strategic investment, particularly in areas of design and architecture.

Recommendation (C-1): *Do not invest in the CMOS Device Platform.* This investment is already being adequately covered by industry. The focus of the IC should be on how to more effectively use the CMOS Device Platform for increasing computational efficiency through better design.

Finding: The imperative to keep costs low will drive industry to mass production of a limited set of standard central processing units (CPUs), memory, and other chips. These choices artificially constrain developer creativity and the computational model.

Recommendation (C-2): *Pursue holistic co-design.* To expand the application design space that CMOS can support, the USG should seek innovation throughout the CMOS technology stack. A holistic co-design paradigm will provide an efficient way to implement new algorithm research. Custom chip architectures may permit surprising efficiency gains, especially when designed along with the algorithm. The IC and USG should plan to leverage industry's scheduled advances but should also invest in expanding the possibilities for CMOS-based designs beyond those that will come from industry.

Finding: Moving bits in CMOS, regardless of origin or destination, is inefficient, and energy costs are not projected to decrease much in the future. Since these costs rapidly increase with distance from the CPU, communication-intensive applications, such as graph algorithms, scale poorly. The von Neumann architecture, where data and program are stored in local memory and shuttled to processor as needed, exacerbates the problem.

Recommendation (C-3): *Break the von Neumann barrier.* The high cost of communication in CMOS has gradually closed off solution options and constrained algorithmic creativity. The USG should go after this problem aggressively, seeking ways to lower the cost to move bits, to move fewer bits, and/or to reduce the distance the bits must travel. Solutions may involve transport technology, machine architecture, and algorithms. Likely, all three will be important, though transport technology is primarily a concern of industry. Algorithm research and subsequent holistic co-design will be the way to select which combinations of technologies to pursue.

Finding: Besides being more energy efficient, superconducting computers are potentially faster, with greater computational efficiency than conventional ones based on a CMOS platform; however, they also present the disadvantage of operating at 4 K. At present, superconducting is lacking a low-energy, high-performance memory, though several approaches appear promising. Many of the architectural principles developed for CMOS are applicable to the superconducting platform.

Recommendation (C-4): *Experimentally demonstrate the operational advantage of the superconducting platform.* Implement superconducting logic in the context of an architecture that can allow for a direct comparison with CMOS. Doing so will require addressing the current deficiencies in cryogenic memory. The Cryogenic Computing Complexity (C3) program of the Intelligence Advanced Research Project Activity (IARPA) is on the right track to implement this recommendation.

Finding: Superconducting logic has the advantage over CMOS for bit transport, since moving bits on superconducting lines entails no energy cost (some cost is still incurred moving bits into the processor from the outside). Communications-intensive algorithms should therefore operate more efficiently in superconducting computers than they would in CMOS-based platforms, where most of the energy is expended moving bits.

Recommendation (C-5): *Explore superconducting communications-intensive machine architectures.* Select several communications-intensive applications of interest to serve as targets for application-specific superconducting machine architectures. Use initial computational efficiency estimates for these designs to drive subsequent technology development.

Finding: Superconducting computers generate significantly less heat than CMOS ones and so can provide a hospitable environment to host three-dimensional (3D) stacked-chip architectures, which are otherwise hard to cool. Achieving the full potential of this technology will require reducing superconducting circuit elements to smaller sizes to enable scaling and developing a novel approach to reliability that can ensure continuous operation of a 3D structure at 4 K.
Recommendation (C-6): *Produce a roadmap for superconducting technology.* The USG should charter a roadmap similar in intent to the ITRS, focused on the critical issues and challenges inherent in scaling to a smaller feature size. Publication of such a document will broadcast serious interest on the part of the government and suggest a way forward to the nascent superconducting community.

Finding: DNA computing has little practical value for tackling NP-hard problems. gates,ally, a brute force technique, the size of the space it can search is limited by the number of DNA molecules that can be simultaneously held in solution. While the technology does offer a significant speed-up over conventional computing for exhaustively searching small spaces, without the ability to scale to larger spaces, interesting problems lie beyond its reach. The molecular-tile approach gives DNA computing a way to implement logic gates; however, it has serious limitations. Instructions execute whenever they are triggered, and there is no easy way to evaluate them in a particular order. Also, to avoid contamination by stray chemical reactions, complex calculations are constrained to proceed very slowly.

Early optimism that molecular computing could offer a general-purpose computing technology using a biological substrate has given way to recognition that it will likely have utility only in a biological context. Most current research is focused on using molecular computing to address computational issues associated with enhancing or adding functionality to a biological system. The biological interactions are slow, occurring on the scale of minutes to days, but they can be exceptionally low power.
Recommendation (C-7): *The IC should not invest in biomolecular technology at this time.* Biomolecular technology is not well suited for hard calculations. Molecular tile calculations are error prone and require slow growth rates to minimize errors. Computation based on biological processes cannot today implement either sophisticated decision logic or instruction sequences. Biomolecular computing will continue to mature without our investment for applications in medicine, fuel production, and environmental engineering. The field should be tracked for developments that might have a potential disruptive impact on general computation.

2.2.2 Neuro-Inspired

Finding: Neuro-inspired hardware computing approaches, despite their promise, are not solving problems of interest. In contrast, neuro-inspired machine learning approaches using software have shown usefulness and scalability, but they require too much digital computation and, therefore, too much compute time and power.

Recommendation (N-1): *Guide hardware development with lessons from machine learning and neuroscience.* Neuro-inspired computing suffers from a lack about understanding of how computation really proceeds in the brain. The USG should sponsor research that uncovers and is informed by the anatomical and biomolecular processes that make the brain so efficient. New neuromorphic hardware design efforts should incorporate findings from machine learning.

Finding: Neuromorphic computing is characterized by simple local processing in the "neurons," whose output affects a nearby subset of the "synapses" that form the application's distributed memory. Data do not move long distances and, therefore, computation has the potential to be extremely energy efficient. Applications in surveillance and robotics may be able to take advantage of this style of computation.

Recommendation (N-2): *Fund hardware research to create energy-efficient "synapses" that can be integrated with CMOS.* Creating this sort of memory architecture may be a stepping stone on the path to building a scalable neuro-inspired processor for larger problems. Current candidate chip-level building blocks are inadequate. Floating-gate structures are too large and require too much energy for updating, and memristors have a high failure rate.

Finding: Neuromorphic computing uses local learning algorithms to update the "synapse" values in response to data patterns at the periphery of the device. Rules based on spike-timing dependent plasticity (STDP) mimic biological synapses and can encode temporal patterns. Machine learning algorithms implemented in software are not always neuromorphic and do not generally account for temporal patterns naturally.

Recommendation (N-3): *Fund research on hardware-friendly temporal learning algorithms.* This research should be done in concert with hardware architecture research while being mindful of the progress being made in neuromorphic-style deep machine learning algorithms. Algorithms, architecture, and representations can draw inspiration from and guide hardware development in a co-design process.

Finding: Biomolecular methods for computing have not yet been used to create neuro-inspired architectures, but the self-assembly properties of DNA may have application to neuro-inspired computational constructs. The field of synthetic biology is advancing rapidly and may also create opportunities for advances in neuro-inspired architectures. Drivers for these technologies will come from the areas of medicine, energy production, and environmental engineering.

Recommendation (N-4): *Watch synthetic biology advances for possible breakthroughs in creating neuro-inspired computers.* Biology may produce potentially disruptive technologies for neuromorphic computation.

2.2.3 Analog

Finding: Analog computing is not well suited to traditional IC problems (like factoring) that require bit-precise calculation or high dynamic range. Analog computing is best suited to problems that allow for approximate solutions and may offer improved energy efficiency and speed over digital computing for some of those problems. It may have a niche role for accelerating the solution of IC problems that can be represented in terms of or modeled by physical systems.
Recommendation (A-1): *Investigate the mapping of IC problems to dynamical systems.* IC problems may have components that can be mapped to dynamical systems and could therefore be accelerated by analog computing. One real-world example is Boolean satisfiability, an optimization technique for which a dynamical systems approach with benefits over traditional algorithms has been demonstrated in principle [44].

2.2.4 Quantum Computing

Finding: Shor's algorithm for factoring integers has served as the overarching motivation for practical quantum computing to date. With the possible exception of quantum simulation, the other known quantum algorithms are primarily of theoretical interest or do not offer a sufficient reduction in calculation complexity to merit investment. Given the theoretical power of a computing model not limited to binary states, the dearth of interesting quantum computing algorithms is surprising and likely results from too much focus on the quantum circuit model. Our attempts to force quantum mechanics into a familiar computing construct may be limiting our ability to understand quantum computation at an intuitive level.
Recommendation (Q-1): *Broaden scope of quantum models of computation and algorithms.* Pursue basic research is aimed at expanding our understanding of computation in quantum systems beyond the digital circuit model. Without being too restrictive, focus attention on real problems where quantum computing can offer substantial improvement over the best classical computing approaches. Widening the applicability of quantum computing will increase industry's interest in sharing a portion of its development cost.

Finding: We have been on a 25-year quest for a logical qubit and are still a long way from achieving that goal. Since quantum states are fragile and protecting them from corruption by the surrounding environment is challenging, preserving computational integrity appears to hinge on effective quantum error correction. Today, some approaches are better than others, but we have no theory to say how much better they can get.
Recommendation (Q-2): *Demonstrate a logical qubit suitable for scalable quantum computation.* Focus efforts towards a practical logical qubit, that is, for which the quantum state can be maintained for the duration of the calculation and that can scale to support large computations of interest. Pursue a two-pronged strategy to

advance understanding of quantum error correction and aggressively seek a candidate physical qubit technology that is likely to meet scalability requirements.

Finding: It is no more likely that we will be able to have built a practical quantum computer from many of today's candidate qubit technologies than we could have built today's high-performance computers from vacuum tubes. The development of a quantum computer cannot be a linear process from qubit to system. We need to better understand the implications on system architecture of candidate qubit systems. To this end, there needs to be a symbiotic effort between qubit technology and system architecture for a scalable quantum computer. Performance stems from achieving balance over the whole system—from qubit to algorithm.

Recommendation (Q-3): *Develop candidate system-level architecture for a capability quantum computer.* A *capability* quantum computer is one that delivers a computational capability of interest to the IC and is scalable. In other words, it must have a useful performance threshold and be readily extensible to larger problems of the same type. This architecture effort needs to be symbiotic with the effort for demonstrating a logical qubit. Our current understanding of the qubit should drive the architecture and the architecture should drive the required performance of the qubit. Research needs to be application specific and consider the scalability of the calculation. The focus should be on achieving balance across the design space from qubit through algorithm.

Table 2.1: This table provides a summary of all the recommendations mentioned in this section. Additional detail can be found in the next chapter.

Computing Style	Device Platform	State	Material	Device	Component Architecture	Node Architecture	System Architecture	Programming Model	Algorithm
Quantum Digital	GaAs Quantum dot	Q-2: Demonstrate a logical qubit			Q-3: Develop candidate system level architectures				
	Si-Phosphorous								
	Si Quantum dot								
	Superconducting								
	Trapped-ion								
Analog	Analog Biomolecular Mixed-signal								
	Analog CMOS Mixed-signal CMOS								A-1: Map problems
Neuro-inspired	Analog Biomolecular Mixed-signal Biomolecular	N-4: IC should watch for disruptive application to computing							
	Analog CMOS Mixed-signal CMOS	N-1: Guide hardware from algorithm and biology N-2: Develop energy-efficient synapse						N-1: Neuro algorithms N-3: Temporal learning	
Classical Digital	Biomolecular	C-7: IC should watch for disruptive application to computing							
	Superconducting	C-4: Demonstrate performance advantage C-6: ITRS-like roadmap for superconducting			C-5: Explore communications-intensive machine architectures				
	Beyond CMOS	C-1: ITRS Challenges and Beyond CMOS			C-2: Pursue holistic design C-3: Break the von Neumann barrier				
	Digital CMOS								

🟨 Industry 🟥 IC Investment 🟧 Other USG

3 The Big Picture

In this chapter, we describe in more detail the "big picture" as we see it. The two chapters that follow this one contain much of the detailed analysis, with references. The final chapter presents several topics that merit further exploration than was possible in the study. These ideas are consistent with the findings and recommendations and may offer a way forward, though we caution the reader that they are preliminary and do not exhaust the possibilities. Because our foremost recommendation is the universal adoption of holistic design for the IC's compute problems, we start with perhaps the best known holistic design example: Google.

3.1 An Example of Holistic Design: Google's Compute Infrastructure

In the early days, Google quickly realized that the scale of computing needed was unprecedented and beyond the reach of commodity solutions. Relational databases were not large enough and were too expensive. In addition, it had to perform a non-standard calculation on a very large graph to determine page rank [90]. Google took ownership of their computing problem and crafted a solution that spanned the spectrum from algorithms to data centers. Its software approach was captured in three applications:

- **Google File System** is a scalable distributed file system for large distributed data-intensive applications. The file system was designed to be distributed over thousands of clusters, backup data through replication, and provide optimized sequential access for large files [50].

- **MapReduce** is a programming model that automatically parallelizes the computation across large-scale clusters and provides reliable operation in the presence of faults. MapReduce was primarily designed for indexing the web and now is part of Google's general computing infrastructure [36].

- **BigTable** is a distributed storage system for managing structured data, and is designed to scale to petabytes. It supports many applications including Google Search and Google Maps [30].

Google designed its own data centers to house compute clusters built from commodity components. On this end of the spectrum, Google made important innovations. For example, it noticed that standard power supplies were much less efficient than they could be and modified them (by removing (AC/DC) conversion stages and installing high-efficiency voltage regulators) to be ∼25% more efficient than typical power supplies. Google's data centers do not have a separate storage area network, as would be the case with a commercial solution; rather all of the storage is contained on hard drives directly connected to processing nodes. A single web query uses hundreds of machines and completes in less than half of a second. The overall system is exceptionally fault tolerant, by design. Google's worldwide enterprise has 19 data centers and consumes 260 MW [52, 54].

Google thoroughly understood its problem and designed and implemented a solution that innovated at the machine architecture and programming model levels, leveraged the commercial market by using primarily commodity components, and could scale to unprecedented size. For example, the metric PageRank was developed to calculate an estimate of the number and quality of links referencing a web site. The iterative algorithm performs the calculations in a manner that can be implemented efficiently on a commodity cluster without expressly defining the graph structure. In addition, by designing its own compute infrastructure, Google was able to take a green energy approach at an early stage of the design process. Without this level of investment and innovation, Google might not be the company it is today.

3.2 Google's Example Applied to the IC

The example of Google's compute needs is not unique (see, for example, the history of D.E. Shaw's **Anton**, a special-purpose computer for studying protein folding). In fact, the example applies directly and indirectly to many government agencies, especially the IC. The Google example is an instance of data analytics—which is an important IC application. Moreover, the concepts embodied in the Google example should drive government agencies and the IC to think out-of-the-box for efficient operations.

While an approach based on commodity hardware is attractive for many reasons, it appears that a conventional CMOS-based digital approach is unlikely to scale to meet the future needs of the IC. The three overarching challenges are

- limiting electrical power consumption through increased computational efficiency,
- decreasing the high energy cost of moving bits, regardless of the origin and destination, and
- achieving application scalability for communication-intensive applications.

Computational efficiency is a measure of the useful work per Watt performed by a computing system. Large systems consume up to 10 MW of power. At a cost of $1 million per year per MW, factors of 2 in energy efficiency have significant operational and performance impacts. Almost all of this power is consumed moving data—on chip, from local and global memory. At present, the energy cost of moving a 64-bit operand between CPU and local memory is 50× the cost to add two floating-point numbers. To move a 64-bit operand between CPU and global memory, the energy cost is 200× (the cost to add the numbers). In 2020 these costs are expected to be 80× and 700×, respectively. The consequence is that communication-intensive

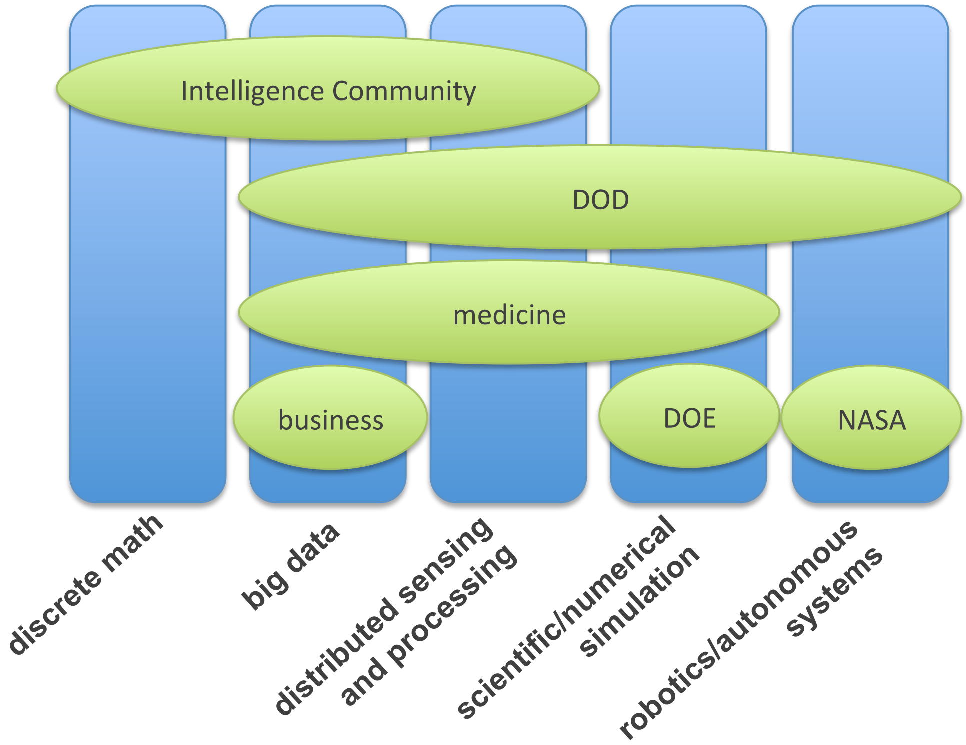

Intelligence Community
DOD
medicine
business
DOE
NASA
discrete math
big data
distributed sensing and processing
scientific/numerical simulation
robotics/autonomous systems

Figure 3.1: Applications of interest to the IC and others in government and the private sector.

applications scale poorly. Although memory bandwidth and access latency are slowly improving with time, they are not keeping up with increases in floating-point and integer operation capacity on a chip. The result is that it is becoming increasingly difficult to keep processors fed with data and instructions from memory [65].

Figure 3.1 shows applications of interest to the IC, the USG, and private sector. The broad application classes are discrete math, big data, distributed sensing and processing, scientific/numerical simulation, robotics/autonomous systems. The first three are of primary interest to the IC.

Within the limits of classical digital computing, we examined superconducting logic (based on Josephson junctions) and biomolecular computing and found that neither of these approaches was capable of overcoming the three challenges across the spectrum of application classes of interest to the IC.

Specifically, while superconducting technology eliminates the cost of moving bits within the superconducting domain, the cost of transferring data from the outside into the superconducting domain *does* dissipate energy at 4 K. Consequently, superconducting is expected to perform poorly on, for example, big data applications where the bulk of data is streamed from the outside into the processor. What is more, the energy cost of logic in superconducting circuits is expected to be similar to that of CMOS.

On applications for which most of the data remain in the cryogenic environment, superconducting is expected to improve computational efficiency by factors of 10–1,000 [61].

The best biomolecular algorithms for classically hard computational problems are still exponential in resources, though, in this case, the exponential resource is molecules and not time. Consequently, these algorithms scale poorly. Molecular tile computing techniques, which expand the possibilities of biomolecular computing, suffer from potentially high error rates. Optimizing the thermodynamics can reduce the error rate but at the expense of excessively slow compute speeds.

We know of no other viable candidate technology for classical digital computation except CMOS and the future evolution of CMOS, which is traditionally termed "Beyond CMOS." The ITRS describes the future of these technologies. Their development is determined by industry, with limited ability for the government to affect direction or timeline. While we did not survey Beyond CMOS technologies in detail, it is clear from the ITRS that there is no silver-bullet replacement for CMOS and that improvements will be evolutionary and incremental. At some time in the future, these incremental and evolutionary improvements may falter and a new disruptive, alternative computing technology will be required to move computing capabilities forward.

Even though CMOS device technology will advance at an incremental and predictable rate, it is less clear that we are getting all we can out of the CMOS device platform. In many cases, the design space is artificially constrained by a host of practical concerns. Software constraints include, for example, compatibility with the programming model. It is constrained at the component level by having to rely excessively on commodity chips because of the expense and time to create application-specific integrated circuits (ASICs). It is also constrained in cost and schedule by by the acquisition cycle. Industry invests 20% of its revenue, which was over $50 billion in 2012, in research and development into silicon material systems. The future is clear—we need to get more out of the CMOS device platform (see Recommendation G-3).

Based on this and other considerations that we will discuss in detail later, we recommend that the IC consider an application-driven, holistic design approach that spans a broad range of technologies and computational models—including the integration of multiple, heterogeneous ACTs into a single system if required. Application-specific designs are not new to either the government or industry. We have described Google's approach. The government—most notably The National Security Agency (NSA) and The Department of Energy (DOE)—have a long track record of investment and innovation to develop solutions for demanding computation problems (see Recommendation G-1).

The classical digital style of computation dominates computing to such an extent that we forget about other computation styles. Table 3.1 lists the four computational styles considered in this report: *classical digital, neuro-inspired, analog*, and *quantum*

Table 3.1: Definition of compute style. Each compute style has a characteristic unit of information and processes to manipulate information.

Compute Style / Model	Unit of Information	Description	Applications
Classical Digital	bit	Information is represented in binary form and manipulated by rule-based logic. Computation not governed by natural laws. Dominant computation style.	General-purpose computation of almost anything. Also used for application-specific computing.
Neuro-inspired	neuron & synapse	Inspired by computational processes in living animals. Processing and memory are fine grained. Connection oriented. Capabilities are learned, not programmed. Computation not governed by natural laws.	Things the brain does well: pattern recognition, discrimination. Potential application to sensor processing and big data processing.
Analog	real variable	Information is represented as a real variable in a dynamical system. Computation governed by the natural laws of the system. Analog computation has limited precision.	Not typically a computation tool. Sometimes used in control loops. Has been used for controlling gunfire and solving differential equations.
Quantum Digital	qubit	Information is represented in qubits in an analog form. Computation occurs through application of quantum gates, which are governed by the laws of quantum mechanics.	Discrete math problems: factoring integers, graph isomorphisms, triangle finding, element distinctiveness. Not an efficient general-purpose computer.

digital. Each style has a basic unit of information and a means of computation on that unit of information. The quantum digital circuit model, for example, represents information in qubits—two-level quantum systems—and information is processed via application of a sequence of quantum gates, which obey the laws of quantum mechanics.

Computational power—that is, the ability to make efficient progress on a problem of interest—is fundamentally related to the style in which the computer computes. Algorithms that are hard for classical digital remain hard for all ACTs and all machine architectures within the class of classical digital computation. Algorithms are intimately tied to computing style. Shor's algorithm for factoring integers is inherently a quantum algorithm and runs efficiently only on a quantum computer. Without an analysis of the alternatives that considers solving challenging problems over the different computing styles, we run the risk of less than optimal solutions or possibly, no solution at all.

A computational style that has the potential for a significant advantage over classical digital but that cannot be reduced to practice results in little utility. Reducing a computational style to practice requires a viable device platform. Figure 3.2 shows the relationship between Computational Style, Device Platform, and Material

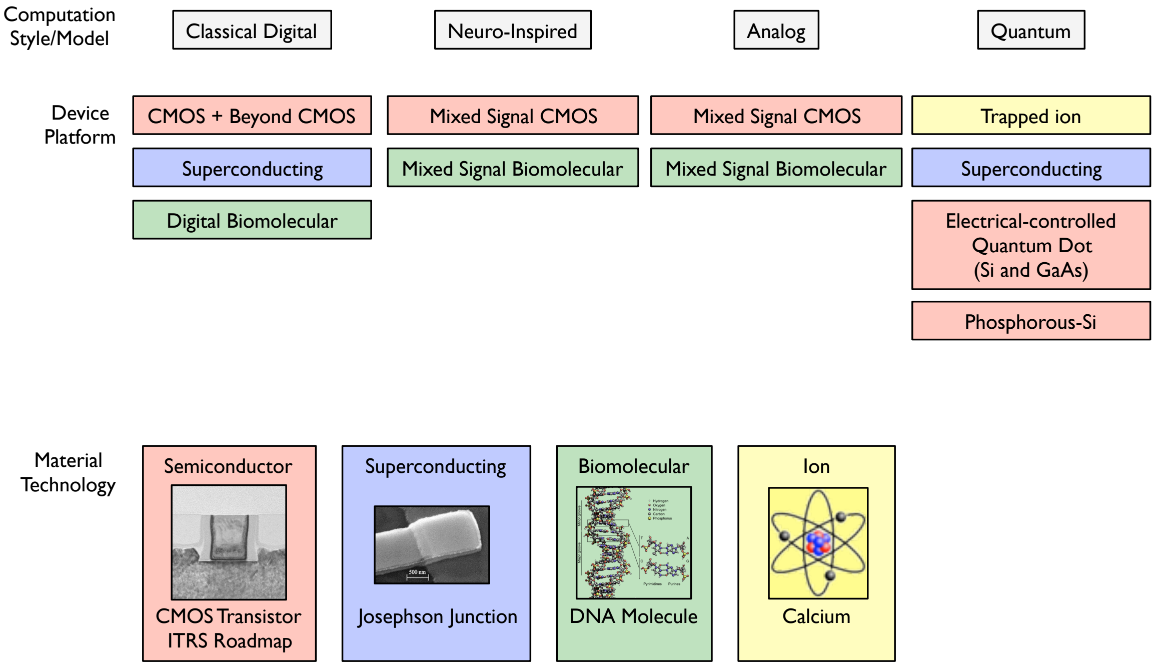


Figure 3.2: Relationship between Computational Style, Platform, and Material Technology. Colors show the relationship between the Material Technology and Device Platform.

Technology. The computational style/model is a theoretical construct that describes how information is represented and how it can be processed. The Material Technology is a broad technology base focused on a particular material system. The four material systems in this figure are Semiconductor, Superconducting, Biomolecular, and Ion. A Device Platform is the technology needed to produce the components (e.g., chips) in the system. Each material technology provides for one or more Device Platforms.

Figure 3.3 shows the organization of a classical digital computing system. The first two layers comprise the compute style and defines the unit of infomation, such as the bit for the digital computation style. The next four layers comprise the Device Platform and provide the capability of producing components for the system. For CMOS, this is the capability of producing CPU, DRAMs, and other packaged computer chips. The next three layers are the Machine Architecture organized as component, compute node, and system. The compute node provides the parts (e.g., chips) to build nodes. The compute node is, in general, capable of standalone execution of the compute model. System level scalability calability is achieved at the system level by interconnecting nodes.

At present, we do not innovate across the entire design space, and, consequently, our designs are suboptimal. Our tendency is to innovate at the node and system level. We build systems mostly out of commodity chips. Innovation at the programming model can be limited because of the need for backward compatibility with an existing software base. Opening up the design space is expensive and may also be prohibited

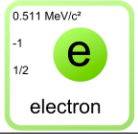
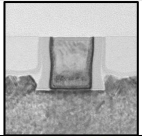
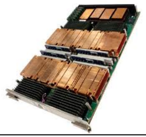
Computational Model/Style	Classical Digital		Style	Defines the intrinsic nature of the computation and the computational unit of information
Computational Unit	Bit	0,1		
State Variable	Charge (electrons)		Device Platform	Physical entity used to represent the computational unit
Material	Silicon			Material system for manipulating the state variable
Device	CMOS transistor DRAM cell			Devices capable of manipulating state
Component	CPU DRAM Co-Processor		Machine Architecture	Component - Computer parts made from the Device Platform
Node	Board von Neumann Architecture			Node - Assembly of components capable of executing compute model
System	Global Interconnect			System - Collection of nodes capable of sharing data
Execution and Programming Model	MPI			Software stack
Algorithm	LINPAC Benchmark	TOP500		Application

Figure 3.3: Breakdown of a system from computational model through algorithm using an HPC computer running LINPACK as an example. The first two layers define the compute style and unit of information. The next four layers (State Variable through Device) comprise the Device Platform, and the next three (Component through System) comprise the Machine Architecture.

because of practical acquisition constraints of schedule and cost. Opening up the design space requires strategic planning and investment. Having to support the entire stack, from state variable to algorithm is expensive but may not be so if extensive leveraging of existing technology is possible. Choices are important and should be grounded in compute style because, from this, stems computational power (see Recommendation G-1).

Developing a Device Platform for which there is not a Material Technology base is very expensive. Consider Semiconductor technology. In excess of $50 billion per year is invested in this base technology, primarily focused on continuing Moore's law for CMOS semiconductors. It is by far the most developed of the four and its future is described in the ITRS. This technology base is larger than just the CMOS device platform. It also includes microelectromechanical systems (MEMS), mixed signal chips, and system on a chip. At present it is technically mature to support all of the compute styles with the exception of Quantum Digital. If a silicon-based device platform is ever developed for quantum computing, it should be an extension of the Semiconductor Material Technology base and should heavily leverage existing capabilities of semiconductor technology (see Recommendation G-3).

Biomolecular Material Technology is not particularly mature for computing. However, the driving applications for biomolecular are pharmaceutics and genomics, not computing. As such, biomolecular has the potential to be a disruptive technology for computing. The remaining two technologies we considered, superconducting and ion, are relatively immature, and have little support from industry, although they partially leverage Semiconductor for fabrication technology.

Figure 3.4 depicts the impact of different approaches to innovation on computational capability. The leftmost plot is the free evolution arising from advances in Device Platform. In the case of CMOS, this is Moore's law driven by the excess of $50 billion annual investment in silicon. It is perhaps slightly misleading to call it free. It is only cost-free when a large user base supports research and development. The upward slope characterizes the increases in compute capability arising purely from advances in the device platform.

The middle plot in Figure 3.4 depicts innovation at the machine architecture level and above. The most effective innovation is algorithmic innovation. Finding a better algorithm leads to a discrete jump in capability, followed, in some cases, by a steeper slope because of more efficient use of computational resources. Novel and optimized machine architectures may also lead to discontinuous improvement. Machine architecture is more costly than algorithm innovation, however.

The rightmost plot in Figure 3.4 shows the impact of a novel alternative computing technology (orange line). In this case, there is a period of essentially *no* compute capability while the novel technology matures. The technology progresses through a scaling phase until it "takes off" and surpasses the conventional approach. The technology clearly needs to scale. For a limited time, the capability can scale simply

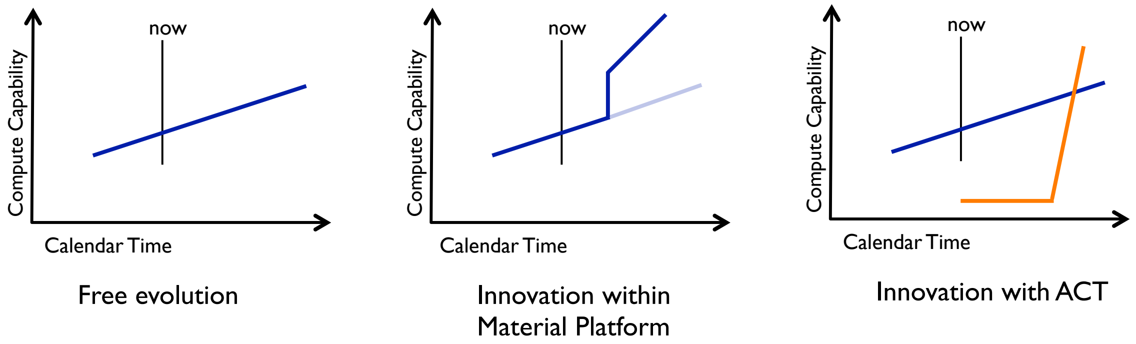


Figure 3.4: Scenarios resulting from different approaches to innovation in computational problems. The leftmost plot represents the "free" evolution in compute capability one obtains by allowing conventional CMOS to improve according to Moore's law due to industry investment. The middle plot represents the possible evolution resulting from directed efforts to innovate at the machine architecture level and above. The rightmost plot represents the evolution of a novel alternative compute technology (orange line) as it incubates and eventually "takes off." For the computational problem of prime factorization, the orange curve might represent quantum computing with Shor's algorithm, while the blue curve represents the evolution of a standard general number field sieve algorithm on CMOS hardware.

by letting the system get larger, but, ultimately, the Device Platform has to improve to provide for device scaling. This capability will have to be paid for by the user base, and it may be possible to offset some costs by leveraging innovations in other material systems. Supporting device level innovation, particularly if the costs of scaling also have to be carried, can rapidly become prohibitively expensive.

These observations and analysis led us to three general conclusions governing the role ACTs should play in the strategic computing landscape. The development effort required to make an ACT competitive may be considerable, and such a high cost must be considered in terms of the potential return on investment. The largest uncertainty is the future value of CMOS, particularly if the USG invests to expand its potential. Industry is willing to invest over $50 billion per year in advancing silicon. The IC needs to leverage this investment to the maximum before resorting to an ACT. Spending upwards of $1 billion per year on expanding the capability of silicon at the machine architecture level merits serious consideration given the cumulative investment in silicon by industry (see Recommendation G-3).

3.2.1 Classical Digital Computing

Classical digital computing represents information as bits and performs a sequence of logical operations on that information to compute. Although physical processes are used to compute, no natural laws govern the computation. Hence, a wide range of potential device platforms is available—everything from the standard CMOS to billiard balls. The three viable Device Platforms we examined were CMOS and Beyond CMOS, Superconducting, and Biomolecular, as shown in Figure 3.2.

In the most common form of the classical style of computing, a stored program controls execution with a von Neumann architecture. The von Neumann architecture stores the program in main memory. Program instructions and data are fetched from memory and executed in the CPU. The von Neumann architecture requires high bandwidth between the CPU and memory. The bandwidth between the CPU and memory often limits performance, and this limitation is called the von Neumann bottleneck. The introduction of multi-core CPUs without a commensurate increase in the memory bandwidth has stressed memory bandwidth even more. There are many variants on the basic von Neumann approach at the machine architecture level, and these systems also support a variety of programming models. The core ideas behind the von Neumann architecture are so ingrained in many of us that we often find it difficult to conceive of computing in any other way.

There is nothing inherently wrong with the von Neumann architecture. The problems stem mostly from limitations at the Device Platform level. Specifically, in the CMOS and Beyond CMOS platform, moving bits dissipates most of the energy, and, by its nature, the von Neumann architecture moves a lot of bits. Alternatively, the superconducting platform moves bits at 4 K, with practically no energy dissipation.

Biomolecular computing is not a von Neumann architecture, though it can have the same compute capabilities, in principle. Information is coded into molecules. Communication is performed by random mixing of molecules in solution, and processing is via chemical reactions driven by heat. Bennett has noted that this is an example of Brownian computation [20]. The computation is a random-walk driven by heat. Dissipation is necessary to prevent the chemical reaction from going in the backward direction. Dissipation of 20 kT[1] (100 kT) gives a probability of error of $e^{-20} \approx 2 \times 10^{-9}$ ($e^{-100} \approx 4 \times 10^{-44}$), respectively. For comparison, CMOS transistors switch at 100,000 kT and superconducting logic is about the same, after accounting for thermal efficiency of the refrigerator need to cool a superconducting computer to 4 K. Hence, biomolecular computing offers the possibility of extremely low-power computation (though one should also include the power required to run the associated laboratory equipment used in the calculation). Table 3.2 presents an overview of our recommendations for the Classical Digital style of computing.

CMOS and Beyond CMOS

A global market for semiconductors of more than $300 billion annually [4] drives more than $50 billion annually [1] into research and development of the Silicon Material System and the CMOS Device Platform. The IC needs to leverage industry's investment to the greatest extent possible.

The ITRS roadmap predicts that the CMOS transistor will reach its practical scaling limit in 2024 and that that limit will be only 3× worse than the theoretical

[1] kT is the thermal energy of a degree of freedom of a system at temperature T. At room temperature (300 K), $kT = 4.1 \times 10^{-21}$ J.

Table 3.2: Recommended investment strategy "map" for CMOS and Beyond CMOS. The technology stack, from state variable to algorithm is listed at the top of each column. Individual device technologies are listed in each row. Our recommended investment strategy for each portion of the map (i.e., each combination of technology and location in the technology stack) is labeled by reference to a particular recommendation from Section 2 in the format C-N.

Computing Style	Device Platform	State	Material	Device	Component Architecture	Node Architecture	System Architecture	Programming Model	Algorithm
Classical Digital	Biomolecular	C-7: IC should watch for disruptive application to computing							
	Superconducting	C-4: Demonstrate performance advantage C-6: ITRS-like roadmap for superconducting			C-5: Explore communications-intensive machine architectures				
	Beyond CMOS	C-1: ITRS Challenges and Beyond CMOS			C-2: Pursue holistic design C-3: Break the von Neumann barrier				
	Digital CMOS								

🟨 Industry 🟥 IC Investment 🟧 Other USG

limit for any charge-based device [131]. Realizing the inevitable end of the scaling of CMOS, the ITRS has put forth the following four challenges to focus research on emerging devices:

- CMOS: Scale CMOS to its practical limit.
- Beyond CMOS: Continue functional scaling of information processing technology substantially beyond that attainable by ultimately scaled CMOS, which may require non-binary data representation and non-Boolean logic.
- Platform: Extend ultimately scaled CMOS as a platform technology. Produce new device technologies and primitive-level architectures to provide special-purpose, optimized functional cores heterogeneously integrable with CMOS.
- Memory: Identify the most promising technical approaches to obtain electrically accessible, high-speed, high-density, low-power, embeddable (with the CPU) volatile and non-volatile random-access memory (RAM).

These challenges, as articulated, provide confidence of sufficient direction and investment from industry at the device level for the foreseeable future. Any investment from the IC would not significantly alter this course. Hence the IC should not strategically invest at the Device Platform Level (see Recommendation C-1).

If the four ITRS challenges are met, the Device Platform in the 2025 time frame (and likely sooner) will offer significantly new and enhanced capabilities. The potential for a new form of embeddable memory opens up interesting architecture possibilities, such as developing new, efficient versions of technologies like processor in memory (PIM). The problem is that bit transport will still dominate the power budget at all levels (chip, main memory, and global memory) remains. Marginal improvement to bit transport technology will help but will be insufficient to change the situation. Left unaddressed, it will impact application performance and scalability. The only avenues remaining to surrmount the problem are via machine architecture and algorithms. Architectures need to move less data and move away from the von Neumann architecture. To improve efficiency,= critical mission applications must involve less data movement. This need will require a holistic co-design effort where new algorithms are developed that minimize data movement and architectures are designed that enable data movement in the most efficient manner. PIM is a natural way of reducing data movement, though these approaches present challenges of their own. We need to fully leverage the capabilities of the Device Platform, particularly given the challenges at the device level that are being addressed by industry. As an example of a distinctly non-von Neumann architecture, and one that is *not* PIM, take the Micron Automata processor [38]. The Automata processor, due to be released commercially in 2014, is an accelerator that computes by directly simulating non-deterministic finite automata in hardware. It does not have an instruction set; rather, its input is a string drawn from an allowed alphabet of symbols. It can be used to accelerate regular expression matching, for example.

Ultimate success will result from tools and techniques for holistic design that encompass the entire design space—from algorithm design proceeding down through the components employed on the chip. This system stack must emphasize using communications efficiently—which ideally will break through the von Neumann wall (see Recommendations C-2 and C-3). As recently pointed out in a JASON study JASON [65], we are unlikely to be able to build a practically useful exascale machine without all-encompassing, holistic codesign. In fact, as evidenced by the presentations at the DOE's codesign "birds of a feather" event at SC13,[2] domain scientists and computational scientists are working together on increasingly holistic codesign efforts, which they call "radical codesign."

Superconducting

Superconducting logic elements use the Josephson junction to generate and process single flux quanta (SFQ) at 4 K. Bits are represented as the presence or absence of SFQ. SFQ propagate ballistically with essentially no propagation loss. There are many variants of SFQ logic, and all can provide a complete set of logic gates.

Systems built from superconducting technologies support standard digital computing and consequently offer a way forward for much of what we know how to do with conventional CMOS computing, from machine architectures to algorithms onto the superconducting technology base, with the potential for several orders of magnitude improvement in performance.

[2] SC13, Denver, CO, November 20, 2013

The primary advantage of digital superconducting electronics is reduced power consumption. Notional systems designs for superconducting exaflop computing are around 1 MW, a factor of 20 less than the DOE exascale target and a factor of 500 less than estimates based on conventional CMOS computing. These estimates assume superconducting logic and memory to have sufficiently low energy dissipation, which has not yet been experimentally verified. (see Recommendation C-4)

The energy required to move bits in superconducting chips is near zero and independent of distance, unlike conventional CMOS computing where moving bits accounts for most of the dissipated power on a chip. This advantage could be very important for computations that are communications intensive and for which the improvements in performance will be greater by moving to a superconducting base technology than they will be for highly local computations involving little data movement. (see Recommendation C-5)

Because the heat generated with digital superconducting circuits is much less than with digital CMOS, many packaging constraints are not as severe. This reduction in thermal load can lead to large and more diverse 3D stacking of chips and consequently an overall reduction in the size of the system (see Recommendation C-6)

Because superconducting systems will rely on advanced 3D packaging, reliability of the components will be key, and systems may require new approaches to reliability. Not only is it difficult to thermally cycle the system for servicing, but it is also unlikely that part of a 3D stack could be serviced beyond replacement of the entire stack. Superconducting systems will therefore need to fail gracefully.

The benefit of superconducting computing may be limited by the scalability of the chip to smaller feature size. Unlike conventional CMOS computing, superconducting logic circuits make extensive use of inductors, which are difficult to make smaller. Ultimately, this may limit scalability and will need to be addressed (see Recommendation C-6).

Processors built from superconducting logic can run faster, which leads to strong scaling, where each processor can do more computation (assuming memory access time is proportional to the increased speed). This will result in smaller systems (less memory) and more efficient computation than under weak scaling where the processor speed does not increase and consequently the size of the problem must increase to keep the amount of work constant as processor count is increased (see Recommendation C-6).

Biomolecular

Over the last 20 years, biomolecular computing has gone through three phases: deoxyribonucleic acid (DNA) computing for hard computational problems, Molecular-tile computing to address computational limitations of DNA computing, and Molecular computing. DNA computing encodes information in oligonucleotides (single strands of DNA), and processing is via Watson-Crick pairing (pairing of DNA bases with their complement) and ligation (the joining of two strands of DNA). DNA computing focused on NP-hard problems, where the many molecules in solution could efficiently explore large spaces. Molecular-tile computing was developed to provide more control over the calculation. Molecular-tile computing uses a set of two-dimensional (2D) DNA tiles that can be thought of as having colored edges. Computation progresses by growing a 2D structure where tiles can bind along edges of the same color. More recently, the focus has been on Molecular computing for controlling biological systems and not computing. Simple logic functions have been demonstrated with a range of molecular mechanism *in vitro* and *in vivo*. Analog computing has also been demonstrated *in vivo*.

DNA computing has been used to solve computationally hard (NP) problems in linear time. DNA computing as a solution for NP problems has limited scalability and consequently has little practical value. The number of DNA molecules that can be contained in solution limits the size of the space that can be searched. While DAA computing does potentially offer a significant speed-up for exhaustively searching small spaces that are beyond the capability of conventional computing, these solutions are of no practical value without an ability to scale to larger spaces. Trading exponential scaling in running time for exponential material resources, as DNA computing does, still leaves the solution to the problem in an exponential space. Instead of running out of time, now one runs out of molecules. The extension of DNA computing to a general Turing-complete machine has been studied but has never been realized. Furthermore, this would not be practical because of the long operational time needed for the sequential steps in typical computations.

Molecular-tile based computing does expand the compute capability of biomolecular computing and provides a basis for universal computation with molecules. However, this general capability has yet to be realized. Algorithms for factoring and Boolean satisfiability still have exponential complexity for spatial resources, and there is no significant advantage over conventional computing for these and other NP problems. This is just the nature of computing with a Turing machine—exponential problems stay exponential. Furthermore, these calculations must proceed slowly to avoid error. Thus, the computation time is excessively long because the growth temperature is near the melting point and molecular-tile concentration is low. Finally, much of the effort in this area has been refocused on the algorithmic nature of self-assembly of 2D and 3D structures, with possible applications to Molecular computing.

More recently, Molecular computing has developed to address computational issues in biological systems for enhancing or adding functionality to a biological system. It is not a general-purpose computing technology using a biological substrate. The biological interactions are slow, occurring on the scale of minutes to days, though they can be exceptionally low power. These techniques are being developed for application in medicine, fuel production, and environmental engineering.

DNA Computing and Molecular-tile computing have severe limitations for computationally hard problems and are not active areas of experimental research. Molecular computing is focused on simple logical operations using a diverse number of mechanisms. The focus is not on computation but on control in biological systems. This research should be funded by other parts of USG (NSF and NIH, for example) and watched by the IC for possible disruptive application to computation (see Recommendation C-7).

3.2.2 Neuro-Inspired Computation

The processing performed in animal brains motivates neuro-inspired computation. No amount of shrinking the transistor will allow conventional digital CMOS computing to do what the human brain is capable of doing with only 15 W. Neuro-inspired computing is characterized by analog and approximate computation with high connectivity. Processing and memory are intertwined. Like the human brain, neuro-inspired computation is expected to be good at stream processing of vision, hearing, and other sensor modalities, as well as online learning. Neuro-inspired processing should also be useful for processing unstructured big data.

Neural-style processing is connection-driven. Information flows along the connections (synthetic axons, synapses, and dendrites) from one synthetic neuron to the next, each of which generally performs a nonlinear transformation on the sum of the input signals to that neuron. Knowledge is encoded in the connection strengths (weights) through a learning procedure. Often, digital spikes are used for capturing temporal events using STDP learning.

There is much we do not understand about how the brain works. While we have a good understanding at the cellular level and at the global organizational level, we lack understanding at the circuit level—how the neurons are organized into interconnected cortical columns, and how these circuits are interconnected. Efforts including the Human Brain Project in Europe and the recent BRAIN Initiative in the United Sates (US) are focused on achieving understanding at these levels. Efforts like these need to continue and even be expanded (see Recommendation N-1). Meaningful progress will require an interdisciplinary team of biologists, control engineers, mathematicians, physicists, and computer scientists. At this point, we do not have a computation model for how the brain works. Developing a rigorous understanding and model of neurological computation should be a key goal.

Neuro-inspired computing can be simulated use a digital computer, implemented in analog CMOS, and could possibly one day utilize analog and digital biomolecular computation in cells. Neuro-inspired computing on conventional digital computers is useful for developing our understanding but, ultimately, because of limited connectivity and low energy efficiency compared to neuromorphic circuits, is inferior for processing across the spectrum, from robotics to big data. Our best choice for realizable neuromorphic computing at present is hybrid analog/digital CMOS.

Analog electronic design is a very mature field and is ready for deployment. However, the extensions needed for scalable Neuro-inspired computation, such as CMOS-ready synapses and the hardware-friendly algorithms, are still in the research

Table 3.3: Recommended investment strategy "map" for Neuro-inspired computation. The technology stack, from state variable to algorithm, is listed at the top of each column. Individual device technologies are listed in each row. Our recommended investment strategy for each portion of the map (i.e., each combination of technology and location in the technology stack) is labeled by reference to a particular recommendation from Section 2 in the format N-X.

Computing Style	Device Platform	State	Material	Device	Component Architecture	Node Architecture	System Architecture	Programming Model	Algorithm
Neuro-inspired	Analog Biomolecular Mixed-signal Biomolecular	N-4: IC should watch for disruptive application to computing							
	Analog CMOS Mixed-signal CMOS	N-1: Guide hardware from algorithm and biology N-2: Develop energy-efficient synapse						N-1: Neuro algorithms N-3: Temporal learning	

🟨 Industry 🟥 IC Investment 🟧 Other USG

phase. Small-scale demonstrations on toy problems exist using current technology (floating gate or large circuit synapses and simple local learning algorithms), but the synapses and algorithms needed for massive streaming data and sophisticated, knowledge-based analyses do not exist.

Analog electronics scales roughly like digital electronics, so the end of CMOS scaling will limit scalability in analog CMOS. However, new devices like memristors and new hardware architectures like 3D and self-assembly can continue the scalability of electronic systems. Neurobiology shows the way to a self-assembled existence proof of scalable, low-power computation. Investment to create the technical breakthroughs in silicon synapses, self-assembly, and scalable temporal learning algorithms can return large dividends in solving real-world problems in robotics and processing unstructured data streams.

CMOS transistors biased below threshold behave like neurons and have been used to create silicon neurons. What is missing for serious neuromorphic computing is a robust and compact synapse technology that can be well integrated with CMOS. The Technology Readiness Level (TRL) of analog CMOS is extremely high for traditional applications but not so for silicon synapses. An effort is needed to develop a high-density, energy-efficient silicon synapse with adjustable weight that is interoperable with silicon neurons (see Recommendation N-2).

Another strong need is a hardware-aware, temporal learning algorithm that can work in large hierarchical networks to create compositional representations. A version of STDP that works on a compact hardware synapse and that can create compositional representations of temporal events and hierarchical sequences would be a major breakthrough. A demonstration of usefulness for a large-scale surveillance problem might be needed to show success. Metrics should include compact size and low power, say a thousand-fold better than digital CMOS (see Recommendation N-3).

Silicon neurons use two to three orders of magnitude less energy than digital neurons (neurons simulated on a conventional digital computer). Biological neurons are seven orders of magnitude more efficient that digital neurons. Analog computation has recently been demonstrated in cell. Ultimately, a synthetic biological platform based on biomolecular interactions may be the most computationally efficient. The area of Molecular computing, with a focus on computation *in vivo* for control in synthetic biological systems, should be watched for potentially disruptive applications to neuro-inspired computing (see Recommendation N-4).

3.2.3 Analog Computing

Analog computing represents information as physical variables (e.g., the voltage on a capacitor) and allows the system to evolve according to dynamical laws (e.g., Kirchhoff's laws for RLC circuits). For an analog system to solve a problem of interest, an analogy or relationship between the analog system and the problem needs to be established.

Analog computing has limited precision (whereas conventional digital computing has unlimited arbitrary precision) and potentially limit its usefulness. The basic operations in analog computing are summation, integration, inversion, multiplication, exponentiation, logarithm, and division.

We did find one hard computational problem of interest to the IC, possibly suitable for analog computing. Boolean satisfiability can be mapped onto a dynamical system described by a set of first order differential equations that could be solved by either a conventional digital computer or possibly by an analog computer. Boolean satisfiability is the canonical NP-complete problem. This approach merits further exploration.

Perhaps what may be more interesting than this mapping of Boolean satisfiability onto a dynamical system is the general notion of mapping problems onto dynamical and solving for their evolution with either analog or digital computation. This merits further exploration (see Recommendation A-1).

Table 3.4: 4 Recommended investment strategy "map" for analog. The technology stack, from state variable to algorithm is listed at the top of each column. Individual device technologies are listed in each row. Our recommended investment strategy for each portion of the map (i.e., each combination of technology and location in the technology stack) is labeled by reference to a particular recommendation from Section 2 in the format A-N.

Computing Style	Device Platform	State	Material	Device	Component Architecture	Node Architecture	System Architecture	Programming Model	Algorithm
Analog	Analog Biomolecular Mixed-signal								
	Analog CMOS Mixed-signal CMOS								A-1: Map problems

🟨 Industry 🟥 IC Investment 🟧 Other USG

Analog computing is readily implemented using the Analog and Mixed Signal CMOS Device Platforms. Analog computation has also been demonstrated using biomolecular chemical reactions in living cells. Decisions on an investment strategy for analog Device Platform and Machine Architecture should be based on first establishing the viability of analog computing to solve an identified problem of interest for the IC.

3.2.4 Quantum Digital

Information in the quantum digital model is represented in qubits (quantum mechanical two-state systems, such as electron spin) and evolves through the application of a sequence of quantum transformations. The archetypal application of quantum computers is Shor's algorithm for factoring numbers. The power of a quantum computer resides in the size of the state space accessible to it, which is exponentially larger than that of a classical digital system. A quantum computer will simultaneously hold more numbers than there are particles in the universe when factoring a 125-bit number via Shor's algorithm. This is possible because of the quantum principle of superposition, which allows a qubit to hold both a 0 and 1 at the same time. Superposition and entanglement give quantum computers computational power that surpasses the computational power of conventional digital computation. Hence, it is not surprising that a quantum computer can reduce the complexity of at least one NP problem from exponential for classical digital computation to polynomial for quantum digital computing.

While the utility of quantum computing for integer factorization (and other applications in that family of problems) is well established, no other quantum algorithms appear to have the same appeal. Why? Is this just a case where the IC mission requirements dominate conventional thinking on quantum algorithms? Are the practical uses for quantum computers that limited? Or, do we not yet have an intuitive way of understanding quantum computation that allows us to fully understand its true power? There are other models of quantum computation, including adiabatic and cluster-state computation. There must be a long-term, sustained effort to understand quantum computation and applications at a deeper level (see Recommendation Q-1).

A practical quantum computer requires a scalable logical qubit. After 25 years of research and exploration of over a dozen candidate qubits, there is still no logical qubit, much less one that is scalable. Scalability is a critical requirement. In the history of computing, technologies that cannot scale do not survive.

Part of the challenge of developing a logical qubit is in protecting the quantum state from the environment. Left unprotected, the quantum state of a qubit survives for about one second, at best. It needs to survive for the duration of the calculation, which can be from hours to days. The logical qubit is the "immortal" qubit. While quantum error correction could protect a qubit from the environment, the overhead imposed by quantum error correction is so great that most of the calculation is spent detecting and correcting errors. The actual calculation is basically insignificant.

Table 3.5: Recommended investment strategy "map" for quantum digital. The technology stack, from state variable to algorithm, is listed at the top of each column. Individual device technologies are listed in each row. Our recommended investment strategy for each portion of the map (i.e., each combination of technology and location in the technology stack) is labeled by reference to a particular recommendation from Section 2 in the format Q-N.

Computing Style	Device Platform	State	Material	Device	Component Architecture	Node Architecture	System Architecture	Programming Model	Algorithm
Quantum Digital	GaAs Quantum dot	Q-2: Demonstrate a logical qubit			Q-3: Develop candidate system level architectures				
	Si-Phosphorous								
	Si Quantum dot								
	Superconducting								
	Trapped-ion								

🟨 Industry 🟥 IC Investment 🟧 Other USG

The five device platforms listed in Table 3.5 are our best chance at present for a scalable qubit. Developing the logical qubit will be expensive and will require at some point a down-selection to one or possibly two approaches. This selection has to be based, in part, on machine architecture considerations. Most of the research to date, however, has been focused on qubit technology. Hence we are recommending a focus on a *scalable* logical qubit (see Recommendation Q-2) and a machine architecture based on candidate qubit technology (see Recommendation Q-3).

3.3 Conclusion

We stress that this study was a broad look at alternative computational technology across a wide class of hard problems of interest to the IC. We did not treat any one problem as being more important than any other problem. Because there was no clear winning ACT across all applications, investment decisions must be made based on priorities of the IC. We made no assumptions regarding priorities of the IC. The one thing that is clear, though, is the importance of computation to the IC. If and when technology advances start to stall at the device level, the advantage will go to organizations that approach problems creatively, and this approach requires a motivated team of creative professionals in a well-supported environment. Adopting this perspective is perhaps the strongest recommendation we can make.

4 Detailed Discussion: Compute Models

We have become a society that makes use of computers throughout our daily lives with little need to understand more than the human–machine interface. Most people have no need to consider the underlying computer and computational science and engineering involved. When considering alternative computing technologies, though, we must consider the computational model explicitly because, ultimately, the capability of a solution is a combination of the computation model and the technology used to realize it. Classical computing models have been used to move the early science of computing forward by providing abstractions to identify designs and other implementations—the Turing Machine and various implementations. Meanwhile, nature also "computes"—and in strikingly different ways from classical computing.

The baseline model for all computing is the classical Turing machine model of computation. Turing-equivalent computers have been realized with vacuum tubes, discrete transistors, silicon-integrated circuits and superconducting logic. We additionally consider three other computation styles: analog, neuro-inspired, and the quantum Turing machine (QTM). As we will see, each model has its capabilities and limitations. Analog computing represents information as variables in a dynamical system and uses the system's dynamical evolution to compute. Neuro-inspired seeks to understand and mimic computational processes in animals. The QTM uses the physical laws of quantum evolution and information encoded into quantum states. Although the classical Turing machine model dominates computing, there are many problems for which other computational models are more powerful. The most obvious example is factoring integers, where the computational power of quantum computing far exceeds that of the classical Turing machine.

There is a substantial gap between the maturity of classical computing and emerging non-classical paradigms. Sustained investment over many years is clearly required to hone the craft into a useable science. It may not be economically possible to advance the rigor of all non-classical approaches, in which case triage is required to determine the appropriate funding level for each approach. We must choose wisely.

4.1 Fundamental Physical Limits on Computation

Computation is inherently physical; however there are several fundamental physical limits on computation are independent of compute model or technology used to implement the model. Before exploring different models of computation in this chapter, or technologies in the next, we describe limits on computation.

4.1.1 Landauer Limit

Perhaps the best known physical limit on computation is the Landauer limit, which states that the erasure of a bit of information is accompanied by an entropy increase of at least $kT \ln 2$, where T is the temperature of the system and k is the Boltzmann constant [70]. (At a temperature of 300 K, this corresponds to 4.1×10^{-21} J.) The consequence is that for irreversible computing (i.e., computation in which information is destroyed) energy must be consumed during computation.

By storing intermediate results, it has been shown that fully reversible computation is theoretically possible [19]. A reversible computer could in principle compute with zero entropy generation and thus zero energy consumption. Although the realization of a truly reversible classical computer is extremely unlikely due to, e.g., thermal noise, reversible computing is an active area of research. Computers built in the reversible style may be much more energy efficient than the conventional, irreversible computers in use today.

Bennett [20] explored the consequences of our inability to isolate computing systems from thermal noise and introduced the concept of the Brownian computer. A Brownian computer is a more realistic form of reversible computer that is driven along its compute path by thermal noise. Bennett showed that it is possible to compute with zero energy consumption, but only in the limit that the computation proceeds infinitesimally slowly. The biological process of DNA replication comes close to realizing the Brownian computer, dissipating $\lesssim 100\ kT$ per step.

4.1.2 Bekenstein Bound

While it might appear at first that some classical physical quantities should be represented by real numbers and, hence, seemingly contain boundless amounts of information in the infinite number of decimal places of such numbers, Bekenstein [16] showed that there is a finite upper limit on the amount of information that can be contained in a finite volume of space with finite entropy. This limit is

$$I_{\max} = \frac{2\pi RE}{\hbar c \ln 2}\,, \tag{4.1}$$

where R is the radius of the spherical volume, E is its energy, and c is the speed of light. Note the presence of the reduced Planck constant, $\hbar$, which hints at the fundamentally quantum nature of this limit. By using the equivalence of mass and energy ($E = mc^2$), we can write the limit as

$$I_{\max} \approx 10^{43} m_{\mathrm{kg}} R_{\mathrm{m}}\,\mathrm{bit}\,, \tag{4.2}$$

where m_{kg} is the mass in kilograms and R_{m} is the radius in meters. The Bekenstein bound on information density is astronomically higher than has been achieved with any computing system built to date [see, for example, 77]. However, the mere existence of the upper limit is sufficient to place constraints on the power of some compute models (see Section 4.3 for one such example).

4.1.3 Bremermann's Limit and the Margolus-Levitin Theorem

Bremermann's limit specifies an upper limit on the computational speed of a self-contained system. This limit is $c^2/h \approx 10^{50}$ bit $\cdot$ s^{-1} $\cdot$ kg^{-1}, where h is the Planck constant. The limit arises by considering the transition between distinguishable states in a physical bit and applying the energy–time form of the Heisenberg uncertainty principle [77]. Bremermann's limit is of practical use in setting unbreakable cryptographic key sizes, for example.

Note that Bremermann's limit is closely related to the Margolus-Levitin theorem, which states that the time required for a system of energy E to move between orthogonal states is $h/4E$ [81]. This limit implies a maximum compute speed of $4/h \approx 6 \times 10^{33}$ operations per second per Joule of energy. The equivalence with Bremermann's limit (modulo factors of order unity) is obtained by taking the energy of a system of mass m to be the rest energy, mc^2.

4.1.4 Efficient Solution of NP-Complete Problems

While it has not been proven, many believe that nature does not efficiently solve NP-complete problems of interesting size [see, for example, 6]. The origin of this belief lies partly in the fundamental limits described previously, which rule out the exact solution of NP-complete problems of sufficiently large size since the number of physical states through which the system must pass is larger than is physically possible by the entire universe. This belief also originates in the observation that when confronted with an NP-complete problem, nature is apparently content with approximate solutions.

Note, however, that this argument refers to the *efficient* (i.e., polynomial time) solution of NP-complete problems. It may be that an alternative, physically based form of computation offers a speedup over conventional digital CMOS for certain types of NP-complete problems while still retaining a super-polynomial scaling with problem size.

4.2 Turing Machine

When we use the term "classical digital computation," as we have done many times, we refer specifically to Turing-complete computation in which information is stored in binary digits. The overwhelming majority of digital computation performed by modern computing hardware (e.g., CMOS-based devices) is of this computational style. As is understood by the reader who has any familiarity with computer programming, computation proceeds in this style as a sequence of algorithmic operations on binary data. For the sake of completeness, we include a brief description of the Turing machine. Note that this is but one instance of a Turing-complete computer, the lambda calculus [122] being perhaps the most famous alternate example.

Turing Machine: In a 1930s paper on computable numbers, Alan Turing presented a description of a hypothetical machine with the ability to compute a "*function*" [124].

> The Turing machine consists of an unlimited amount of memory (in the form of an infinite tape marked off into sections) where each section contains a symbol. Only a single symbol is entered into the machine at any time. The machine can alter the scanned symbol and the machine behavior in part due to the particular symbol and a table of rules. No other symbol on the tape affects the behavior of the machine. The tape can be moved forward and backward so that any symbol can have input to the machine.

The Turing Machine [124] is of note because it specifies a very limited number of atomic operations that can be used to construct more complex procedures, providing the basis for computing algorithms. The Turing Machine is a very simple concept, but it has the capabilities of any modern digital computer. Thus, we tend to think of the Turing Machine as a computer with a processor and read/write memory that implements computer codes/programs. This perception is generally incorrect, because the write process is limited to the read location.

von Neumann/Princeton Paradigm: The compute model conceived by John von Neumann differs from the Turing Model by permitting RAM and more operations/capabilities in the processor. Any operation can read from or write to any memory location independently from the previous operation. The processor has a set of built-in instructions that include binary arithmetic and conditional branching. Contents of memory can be either instructions or data, both residing in the same memory address space. The instruction set is static and the data moves to the instructions. The von Neumann model can compute the same functions as a Turing Machine—thus offering a subset of capabilities. The von Neumann architecture is limited to sequential processing; thus parallel computing architectures are referred to as non-von Neumann architectures.

Harvard Paradigm: The Harvard compute model [58] has physically separate storage and interconnections to memory for program instructions and data. This architecture permits (1) optimal storage of different sized instructions and data and (2) the ability to access an instruction and data concurrently. These two differences set the Harvard paradigm apart from the von Neumann paradigm.

Algorithmic Paradigm: The algorithmic paradigm models the system as a black box, isolated from external influences. A program maps initial inputs to a final output via a deterministic function. Consequently, random behavior or results are undesirable. Also, it is assumed that the function has a beginning and that end and the processor can be switched on/off. [119]

Refinement Paradigm: A specification of the algorithm exists and can be refined to a provably correct code through a series of incremental transformation steps. Answers are logically true/false and provably correct. Thus, binary representations work well. [119]

Pure Logic Paradigm: Only the code/algorithm matters since the computing hardware and the hardware implementation are irrelevant. This paradigm has provided codes that can be run on various processors by transforming the logic into a series of instructions for the native processor instruction set [119].

Stochastic Computing Paradigm: The paradigm is a method of computing where continuous values are represented by streams of random bits. Complex computations are performed by simple bit-wise operations on the streams. Modern digital computing has overcome the computational issues with highly accurate specialized arithmetic engines. [8]

Probabilistic Computing Paradigm: The paradigm method of computing where the output of the circuit is correct with some probability p, where $p < 1$. Probabilistic computing is most applicable to applications that either require or can tolerate non-deterministic or probabilistic behavior. [82].

4.2.1 Technologies

Three classes of computing devices/technologies follow one or more of the classical models: (1) CMOS and Beyond CMOS, (2) Cryogenic Superconducting, and (3) Biomolecular. CMOS and Beyond CMOS and Cryogenic Superconducting clearly are classical digital devices/technologies and follow classical paradigms. One obvious deviation from the Turing machine and the von Neumann or Harvard paradigms is that they will be highly parallel technologies. Meanwhile, they clearly follow the Algorithmic, Refinement, and Pure Logic paradigms. Biomolecular computing offers two possible ways to follow classical paradigms:

- Implementation of synthetic genetic circuits that use recombinases to implement Boolean (digital) logic functions (Algorithmic, Refinement, and Pure Logic Paradigms)

- Using proteins and enzymes with highly predictable responses to implement functions (Turing Machine).

4.2.2 Algorithms and Processing

Classical computing models all rely on the Algorithmic and Refinement Paradigms described previously. Characteristics of determinism and provably correct answers that are definitely true or false are considered inherent to computing as we know it today. Binary logic and arithmetic are intended to provide a belief that the computer provides correct answers. Meanwhile, we may not be able to rely on such characteristics in alternative computing technologies or even in near-future CMOS-based designs that rely on very low power. In such cases, new models of Stochastic or Probabilistic computing may require changes in the way that we compute, the ways that we develop algorithms, and the way that we use the answers from these technologies. Stochastic computing once was examined as a low-cost alternative to conventional computing. Current/future technologies may have uncertainty in the underlying hardware operations, and stochastic/probabilistic computing may provide a way to deal with the uncertainty that occurs because of smaller CMOS feature sizes and reduced power. It may be possible to implement abstractions that incorporate non-determinism to provide greater reliability in massively parallel, fault-tolerant machines.

4.3 Analog

4.3.1 Theoretical Introduction

At its core, the computational model for analog computing is the analogy. One "computes" by manipulating and studying a surrogate system whose structure and evolution are analogous to the system being modeled. For example, since the second-order ordinary differential equation (ODE) that governs the evolution of a mass-spring-damper system is analogous that of an RLC circuit, an RLC circuit can be used to study the response of a mass-spring-damper system. With the introduction of the operational amplifier in the 1940s, generalized analog computers that could solve sets of differential equations of high complexity were designed and built [see, for example, 79]. As recently as 60 years ago, analog computers found widespread use in a number of applications, including process control in industrial production and in ballistics and fire control systems.

Analog computing is sufficiently versatile that the concept of a general purpose analog computer (GPAC) has been developed and studied [112]. It has been shown that the output of a GPAC is the solution of an algebraic ordinary differential equation (ODE) initial value problem [104]. The GPAC concept was extended by Rubel [105], who showed that the extended analog computer (EAC) could also solve initial-value and boundary-value problems defined by partial differential equations. While the EAC originated as a theoretical construct, it should be noted that an EAC has apparently been realized in hardware [84].

It is natural to ask: Are there limits to the computational power of analog computers? While the theory of analog computing has been studied in some detail, progress in our understanding has not matched that of digital computing. What distinguishes analog from digital computing is that an analog computer operates in a continuous state space over continuous or discrete time, whereas a digital computer operates in a discrete state space over discrete time. As one might expect, then, analog computing has been modeled as computing over the real numbers (in contrast to digital computation over discrete numbers) [21]. Branicky [24] showed that a Turing machine could be simulated by a set of ODEs, suggesting that analog computing has at least the power of a Turing machine. Siegelman [116] studied the theory of analog computing in great detail and showed that the recurrent neural network (a type of sequential-time analog computer operating over real numbers) has super-Turing power. However, as has been pointed out [see, for example, 79], this super-Turing power appears to originate in the system's access to non-rational real numbers of essentially infinite precision, which contain infinite information. Super-Turing analog computing is therefore physically unattainable because the Bekenstein bound limits the amount of information that can be contained in a given volume of space [77].

Another fundamental physical limit is especially relevant for analog computers—the Planck scale. In many analog computers, numbers are represented in unary format. This representation may not be obvious at first inspection, since analog computers appear to operate on real numbers with precision limited by noise, for example. However, at the most fundamental level, numbers are represented by repeated quanta at the Planck scale [see, for example, 64]. While this distinction is not particularly useful for typical analog applications (e.g., approximate solution of ODEs), it is important when considering the applicability of analog computing to problems requiring precise operations.

For an example problem requiring bit-exact operations, consider the computational number theory problem of prime factorization. In prime factorization, there is no sense of an approximate solution. In other words, the prime factorization of a number $N+1$ is, in general, unrelated to the prime factorization of N. To perform prime factorization via analog computing thus requires that the representation of an integer has sufficient precision to represent all digits of the integer. For an integer N represented by a physical quantity x, we thus require

$$\frac{\delta x}{x} \leq \frac{1}{N}\,, \tag{4.3}$$

where δx is the precision of x. In other words, the number of significant figures in the measurement of x must be greater than the number of digits of N. The most precise spatial and temporal measurement devices achieve measurement precision of $\delta x/x \sim 10^{-23}$ (for Advanced Laser Interferometer Gravitational-Wave Observatory (LIGO)) and $\delta t/t \sim 10^{-14}$ (for atomic clocks). The physical maximum achievable precision is set by taking the entire universe as our analog computer and comparing with the Planck scale, which is the smallest measurable scale. The Planck length is $(G\hbar/c^3)^{1/2} \approx 10^{-35}\,\mathrm{m}$, while the size of the observable universe is $\approx 10^{27}\,\mathrm{m}$. This fundamental limit on length implies a maximum accuracy on distance measurement of $\approx 10^{-62}$, or 62 decimal digits. The Planck time is $(G\hbar/c^5)^{1/2} \approx 10^{-44}\,\mathrm{s}$, while the age of the universe is $\approx 10^{17}\,\mathrm{s}$. This fundamental limit on time implies a maximum accuracy of time measurement of $\approx 10^{-61}$, or 61 decimal digits. Other physical quantities (e.g., mass, charge, voltage) are similarly constrained. The upshot is that analog computing appears unable to represent (and thus perform operations on) numbers with more than ~ 62 digits. Depending on the application, number theoretical calculations on numbers of this size might be considered “toy” problems.

The preceding argument assumes pure analog computing. Note that hybrid approaches that represent numbers more efficiently are possible [106] and may form the basis of analog computing machines that are more appropriate for problems requiring an exact solution.

4.3.2 Practical Introduction

The preceding discussion highlights some of the theoretical aspects of analog computing and points out how it is ill-suited to certain problems requiring exact solutions. In this section, we describe the applications that are well-suited to analog approaches and discuss the many potential benefits of analog.

In the early days of electronic computing, many companies were producing analog computers capable of performing programmable calculations. With the advent of the digital computer, most companies producing analog computers were gone by 1970. This class of analog computer disappeared because it solved only a narrow set of problems. As that set of problems increased, so did the complexity of the problems and the complexity of the analog computers. The increased complexity of the problems resulted in the system becoming more difficult to program. For systems performing calculations, the marketplace very quickly made its choice, and the choice was digital computing for those applications.

Analog computing, after having been displaced by the rise of digital computing, is being considered seriously again. The primary motivating factors are the high energy efficiency of analog arithmetic operations and the high spatial density of analog devices. For example, analog devices have been shown to solve ODEs faster and with fewer transistors and much less energy than achievable with digital computation.

Computation in analog CMOS often exhibits a physical style where, for example, summation is effected by Kirchoff’s Law for current addition and multiplication is effected by Ohm’s law for the voltage drop across a resistor. These structures take up much less space than the equivalent digital adders and multipliers and, if the currents are small, will dissipate far less power. The precision of such arithmetic operations is much less than the 32 or 64 bits in digital CMOS, but it may be sufficient in an adaptive analog neuromorphic system where a learning algorithm provides the feedback needed to adjust for device mismatches and imperfections.

With the looming end of Moore’s law, analog computation, especially in CMOS technology, is receiving a fresh look, particularly for neuromorphic devices. As mentioned in Section 4.4, there is a need for a compact synapse which can be integrated into an analog CMOS technology. Floating gates are an example of one that works, but it is important that they be able to implement machine learning algorithms at low energy.

One important practical consideration in building analog devices is system design. Analog computers require much greater attention to design and calibration than digital devices. Similar considerations apply when discussing neuromorphic design which is a hybrid of digital design, for spikes and possibly weight storage and analog design for non-linear neurons, synapses, and connections. However, the possibility of using adaptation by learning algorithms may solve the problems of low precision and calibration tuning.

4.4 Neuro-Inspired

4.4.1 Description

If the von Neumann style of processing is characterized by digital and logical computation with limited fanout across digital logic, separation of memory and processing with a limited bandwidth communication channel for access, and stored program control of the output, the pure neuromorphic style of processing is the polar opposite. In the extreme, it is characterized by analog and approximate computation with high fanout, where memory and processing are intertwined as the computation proceeds due to a high bandwidth web of communication channels and where control of the output is directed as a result of learning. Between these extremes lies a rich range of possibilities for hybrid computation. Most neural network machine learning implementations use a digital computer in which the neuromorphic architecture is simulated. Also, many hardware implementations of neural network processors in analog CMOS use digital pulses for communication and may store memories in digital form. This hybrid nature reflects the preeminence and ubiquity of CMOS digital processing today and the ease with which one can design working systems using CMOS.

The types of data processing problems best suited to this style of computing are those that humans do better than von Neumann computers. This work includes many natural, unstructured “big data” processing tasks such as data stream processing for vision, hearing, and other integrated sensory modalities rather than pre-assembled “batch” structured data processing such as that performed on databases. Typically, the organization of a neuromorphic system for processing data and creating inferences is created mostly by machine learning algorithms rather than strictly by rule-based logic programming that is common for structured databases.

The great proliferation of machine learning algorithms and the continual progress in improving inference and analysis indicate that machine learning is not yet a mature field, and, therefore, any neuromorphic hardware should have some amount of configurability and programmability to be flexible enough to incorporate new, improved algorithms. These deficiencies imply that one should study the computational primitives associated with machine learning and neuromorphic processing to be sure that the hardware design allows for the inclusion of all primitives in a configurable system while still offering energy efficiency and computational advantages.

Neuromorphic design is often characterized by dynamic analog ensembles that are massively interconnected, with learning and reconfiguration locally calculated so that there is limited need to move data long distances. This dependence on local computation largely avoids the high energy expenditure associated with long-distance memory accesses that is common in high-performance computing.

The intelligence and defense applications suited for this style of computation include audio-visual surveillance, robotics, and semi-autonomous remote systems that may include sensors and actuators. In addition, streaming data from web videos, news sources, social networks, and other “big data” streams may be monitored with intelligent neuromorphic processing in a semi-autonomous manner.

4.4.2 Algorithms/Processing

Neural-style processing is connection driven. Information flows along the connections (synthetic axons, synapses, and dendrites) from one synthetic neuron to the next, each of which generally performs a nonlinear transformation on the sum of input signal to that neuron. “Knowledge” is encoded in the connection strengths (weights) through a learning procedure, such as the Boltzmann machine algorithm [9], which adjusts the values of the weights according to patterns at the input and output of the neural network. Often, digital spikes are used for capturing temporal events using STDP [97].

Neuromorphic processing can be performed using many different technologies. Technology-specific discussion of neuromorphic computing styles can be found in Sections 5.1, 5.3, and 5.4.

4.5 Quantum Turing Machine (QTM)

By the early 1980s it was known that due to the inherent nature of quantum mechanics, classical computers could not efficiently simulate quantum systems. Quantum computing originated with Feynman [45] as a means to naturally and efficiently simulate physical systems. In pursuit of understanding the power of quantum computers for more general computation, Deutsch [37] proposed a physical basis for the Church-Turing thesis.[3] Deutsch’s treatment showed that the QTM[4] is inherently more powerful than the classical Turing machine for certain types of problems. In particular, QTMs make use of the quantum mechanical phenomena of superposition and entanglement, which, for example, allow for “quantum parallelism.” Quantum computers are also the perfect “analog” for quantum physics calculations, since they offer access to exponentially sized state spaces and are governed by the same physical laws.

Perhaps the most famous application that could benefit from a quantum computer is prime factorization. On a classical digital computer, the best known algorithm for

[3] The Church-Turing thesis states that functions are algorithmically computable if and only if they are computable by a Turing machine.

[4] The term “universal quantum computer” is also used.

factoring scales superpolynomially in the size of the number to be factored. However, with Shor's algorithm, factorization requires a number of operations on a quantum computer that polynomially in the size of the input number. For sufficiently large problem sizes, quantum computing and Shor's algorithm could obviate the need for classical factoring. However, Shor's algorithm and variants thereof can be used for more than simply prime factorization. Shor's algorithm can also be used for the efficient solution of the discrete logarithm problem,[5] and could therefore be used to break the vast majority of public key cryptosystems in use (including, for example, those based on elliptic curves). Although quantum computers that could attack real-world problem sizes are far from being realized in the lab, cryptographers perceive the threat as sufficiently worrisome that the study of "post-quantum cryptography" is growing.

Since the IC is very familiar with the quantum model of computing, we forgo more detailed discussion of the QTM. However, we make one point that we think is particularly important. The power of quantum computing, like any computational model, is intimately related to the algorithms designed for it. While we currently know of one "killer" algorithm in Shor's and a few other potentially useful algorithms (e.g., Grover's), quantum computing is so recent in modern history and quantum mechanics is so foreign to our intuition that many more useful algorithms may be waiting to be uncovered. While the reasons we have not made more progress in uncovering such algorithms have been hypothesized [115], we simply note that continued investment in *understanding* quantum computing is clearly warranted.

[5] More generally speaking, the quantum period-finding algorithm on which Shor's algorithm is based solves the hidden subgroup problem over finite Abelian groups, of which prime factorization and the discrete logarithm problem are two specific examples [see, for example, 66].

5 Detailed Discussion: Compute Technologies

This section considers the technology base used to practically realize a computation model. These five technology bases are Digital CMOS, Superconducting, Analog CMOS, Biomolecular, and Quantum Technologies. Each section presents Findings that mostly describe key characteristics including capabilities and limitations of this technology.

5.1 CMOS and Beyond CMOS for Conventional CMOS Computing

CMOS technology is used for digital integrated circuits, analog circuits, MEMS, and integrated transceivers for communications. Patented in 1963, CMOS is ubiquitous today and is anticipated to extend into the future—how far is uncertain. Conventional CMOS computing uses this technology to implement microprocessors, memory, controllers, and network interfaces that are components in modern computer architectures. CMOS technology has been used extensively for over 30 years mainly due to significant evolution over time. Feature size of transistors and wires has shrunk from 10,000 nm[6] to 22 nm. The number of transistors per processor has increased significantly due to the smaller sizes of transistors, denser packaging, and larger wafer sizes. It is anticipated that feature size will decrease to 14–16 nm in 2014 and less than 10 nm by 2020, approaching 5–7 nm in 2025. The lowest-level components used to develop devices have changed more than in just size, as new implementations have been needed to overcome problems as transistor size shrinks. In the past, computational capability increased by increasing the clock speed—more work was done sequentially per unit time. This increase in clock speed is no longer the case. Clock speed for CMOS has largely stalled and the additional transistors are used to increase the number of processing cores on the CPU.

Historically, the development period between the choice of a research technology and initial production is usually 24 months. Wafer production ramps up to full scale volume in 24 months after initial production. Thus, it takes only 4 years from a research proof-of-concept design of a new transistor to full production. With this rapid research to production cycle, it is difficult to predict the specifics of future CMOS logic devices. It is anticipated that by 2020, Beyond CMOS capabilities will be available to augment the capabilities of CMOS. Thus, the baseline device construction technology against which all others will be compared will be either CMOS or Beyond CMOS depending on the timeframe.

The ITRS examines the state of semiconductor technologies and emerging research devices to assist in directing research and development to move this field forward. It is

[6] The Intel 4004, the first integrated circuit microprocessor, had a linewidth of 10 μm [2].

important to note that many areas are driving semiconductor research, including radio frequency, analog, mixed-signal, and MEMS. The commercial sector will provide a substantial portion of the research and development funding for the technologies used to further CMOS and Beyond CMOS technologies. Meanwhile, the commercial sector may have future requirements that do not meet the needs of the HPC community. The needs for low-power consumption in a handheld, end-user device may drive the research to different technologies than those required for massive calculations in HPC.

5.1.1 Technologies

One of the most significant drivers in feature size reduction will be to find new materials and system designs that will reduce power consumption. CMOS replaced N-type metal-oxide-semiconductor (NMOS) logic because CMOS logic dissipates less power, only requiring power when dynamically switching. As CMOS feature sizes have shrunk, transistor switching times have slowed and new forms of NMOS transistors (that have leakage voltages approaching CMOS dynamical switching) have been used to reverse this trend and increase switching speeds. It will be imperative that new generations of High-K dielectric materials are found, along with new materials to replace stressed silicon. New electrostatic control structures such as multi-gate on enhanced silicon on insulator (SOI) will be required. Delays in any of the supporting technologies may provide enhanced opportunities for exotic Beyond-CMOS technologies.

The Beyond CMOS community is very conscious of the issue to reduce power, while providing the levels of performance needed for some applications. The extreme power reduction community is examining low-power tunnel field-effect transistors (FETs). The device performance community is examining new technologies to augment CMOS with faster, more efficient transistors (e.g., the use of carbon nanotubes).

In the interim, technology packaging may offer ways to attack the power consumption of processing systems. Significant research investment into PIM technologies has demonstrated the utility of the concept, but the technology has proven difficult to build because DRAM manufacturing technologies have proven ill suited to build processor components. Another option being examined is stacked memory integrated with a processor, or processor-under-memory (PUM) technology. Currently, this technology has encountered problems with dissipating heat generated by the processor. Another packaging solution may be to expand the system-on-a-chip concept to the wafer-scale with multiple processors tightly integrated with 3D stacked memory. The interconnection network could be embedded in the wafer, with routing technology integrated onto the wafer. This concept may be able to reduce racks of equipment to the individual wafer scale, significantly reducing power consumption and improving performance

5.1.2 Conventional Algorithms and Processing

For HPC, CMOS is considered synonymous with digital computing. Issues with reducing power may require stochastic or probabilistic computing algorithms to provide answers with predetermined accuracy if the power reductions cause less reliable operations. Holistic design will be significant as CMOS and Beyond CMOS technologies move into the future. It will be important to design systems that make the most power-efficient use of available processors, memory, and interconnection networks while optimizing performance on specific applications of interest. The fact that few or no changes will be required in algorithms or programming models, also supports the desire for CMOS and Beyond CMOS to be the baseline device platform against which all others will be compared.

5.2 Digital Cryogenic Superconducting

Cryogenic Superconducting computing may offer an attractive low-power alternative to CMOS with many additional potential advantages. Josephson junctions provide for devices with rapid switching, permit processor clock rates of 100+ GHz, dissipate little energy per state change, and communicate information via small, nearly lossless current pulses. Past research into the components of cryogenic superconducting computers were met with significant technical obstacles that prevented serious exploration of superconducting computing. However, recent research has provided innovations that create the foundations for a major breakthrough. New single flux quantum (SFQ) logic circuit technology has no static power dissipation, reducing power and heat. New energy-efficient cryogenic memory will permit the implementation of complete HPC systems within the cryostat, including both cryogenic processors and memory.

Studies indicate that superconducting supercomputers may be capable of providing 1 PFLOPS[7] (or 1 PIOPS) for about 25 kW and 100 PFLOPS (or 100 PIOPS) for about 200 kW, including the power to operate the cryogenic refrigerator [61]. Power budgets for systems based on commercial, 60 Hz cryogenic refrigerators are estimated to be 67 GFLOPS/J for a 1 PFLOPS system, improving to 500 GFLOPS/J for systems 100×–1,000× the computational performance of conventional supercomputers. Refrigerator efficiency improves considerably over this range and provides a significant contribution to reach system efficiency goals with 100–1,000 PFLOPS systems. Refrigeration systems are commercially available, with the product driving the market being the use of superconducting magnets in Magnetic Resonance Imaging (MRI) systems.

Cryogenic superconducting computing research has been limited to developing system components with no initiative focused on integrating components into a system. As an essential first step to reduce risk, IARPA's C3 Program will be an attempt to develop proof-of-concept technologies integrated into a system at smaller scales before any attempt to build a superconducting system at scale. The C3 program

[7] A FLOPS is a floating point operation per second. 10^{15} FLOPS is a peta-FLOPS (PFLOPS), and 10^{9} FLOPS is a giga-FLOPS (GFLOPS). Similarly, an IOPS is an instruction operation per second.

will address the challenges of providing sufficient amounts of amply fast cryogenic superconducting memory, adequate integration density, and a realization of complete superconducting HPC systems. The C3 Program will address these challenges with the goal of establishing superconducting computing as a long-term solution to the power-heat dissipation problem. Thus, cryogenic superconducting technologies may challenge CMOS and Beyond CMOS for power-efficient HPC at scale. The success of C3 may pave the way for a new generation of superconducting computers that are scalable, useful for a wide range of user applications, and far more energy efficient than end-of-roadmap CMOS.

5.2.1 Technologies

The state-of-the-art research in cryogenic superconducting computing is currently using niobium as the superconducting material with which to build the SFQ Josephson junctions. As research progresses, other materials may provide better feature scaling or performance. The field has to be expanded to learn more about a broad range of materials and device components. With advances in cryogenic refrigerator technology, some advanced CMOS and Beyond CMOS technologies may become so sufficiently energy efficient at low temperatures that they may become viable technologies while being adequately low energy. Significant advantages may be possible if the capability exists to run similar Beyond CMOS technologies at room temperature for small systems and at 4 K for systems at scale. Such systems could be holistically designed for particular algorithms. Design work could progress on small systems, while applications at scale would run on systems with greatly reduced power consumption when compared to scaling systems of similar materials at room temperatures.

Some packaging technologies for CMOS devices may encounter heat dissipation problems that may be readily surmountable in a superconducting environment. It may be possible to implement packaging designs such as processors under (stacked) memory in a cryogenic environment where heat dissipation may make such packaging unusable at room temperature. If the materials have superconducting characteristics, very little heat may be generated, providing the utility of large quantities of memory residing close to very fast processors.

With such possibilities to provide low-power HPC at scale, providing adequate funding to grow the superconducting computing community may be required. While potential advantages for HPC at scale are significant, the advances will be funded by the commercial community. The current drivers for low power devices, mobile computing and communications, will have little impact on the cryogenic superconducting computing community. Thus the USG may have to shoulder the responsibility to fund research and development in this area.

5.2.2 Algorithms/Processing

Because of similarities in programming models and general similarities in algorithms between conventional CMOS and cryogenic computing technologies, it is simple to compare requirements of superconducting technologies with previous activities like the Defense Advanced Research Projects Agency (DARPA) Exascale Report and the recent JASON study on Exascale Computing for DOE. This comparison is only a first-order estimate because any new system—including a cryogenic superconducting-based system—should be designed in a holistic co-design manner that will likely modify the algorithms and architectures and the technology in the future as research progresses.

The DARPA Exascale study assumed that the 1,000× performance improvement needed to move from Petascale to Exascale would come entirely from incorporating additional processors into the machine, which would further compound programing challenges with the additional threads and concurrency. Meanwhile, superconducting technology will offer faster processors that may enable easier scaling to extreme levels such as Exascale. It was assumed that with conventional CMOS, faster processors would cause issues with power consumption and would stress the memory system. Thus, it was assumed that processor speed would remain basically constant. All scaling would come from weak scaling[8] where problem size would need to be increased to keep the amount of work constant as additional processors were assigned to the task. Increased problem size affects the amount of memory, the size of the interconnection network, and the amount of intermediate storage. Scaling of computation size depends on the application, but the sweet spot in memory size assumed a scaling where memory increased by a factor of approximately 100× vs. 1,000× because of the constraint on processor speed. Meanwhile, with faster processors (if memory access time is proportional to the increased speed), less of the 1,000× performance improvement would need to come from weak scaling. For a 100GHz processor, the memory size increase would be just 11.7×, reducing the amount of memory from 50 PB to under 6 PB. Similar reductions in interconnection network and intermediate storage would have an effect of reducing power consumption for the entire machine, likely by an additional order of magnitude over standard (room temperature) CMOS.

Cryogenic superconducting will support standard digital computing, thus likely requiring few or no changes in algorithms or programming models. It offers a way forward where much of conventional CMOS computing has equivalent and similar counterparts in superconducting technology but offer the potential for several orders of improvement in performance (reduction in electrical power and computation time).

[8] Weak scaling and strong scaling refer to the following behavior when dedicating more compute resources (e.g., nodes in a cluster) to a problem of a given size. In *strong scaling*, the problem size is fixed and the additional compute resources allow the calculation to be completed more quickly. In *weak scaling*, the amount of work per compute element is fixed, and the additional compute resources allow for a larger calculation size. As discussed in the text, it is likely that some problems which are currently in the weak scaling regime on current commodity hardware will be in the strong scaling regime on future alternate computing platforms (e.g., superconducting). The strong scaling regime is preferable since it can be used to reduce calculation time.

Holistic design will be significant as cryogenic superconducting technologies move into the future. It will be important to design systems that make the most power efficient use of available processors, memory, and interconnection networks while optimizing performance on specific applications of interest.

5.3 Analog CMOS

Analog electronic design is a very mature field and is ready for deployment. However, the additions needed for scalable neuromorphic computation such as CMOS-ready synapses and hardware-friendly algorithms are still in the research phase. Small scale demonstrations on toy problems exist given current technology (floating gate or large circuit synapses and simple local learning algorithms) but the synapses and algorithms needed for massive streaming data and sophisticated, knowledge-based analyses do not exist. Analog electronics scales roughly like digital electronics so the end of Moore's law will limit scalability in analog CMOS. However, new devices such as memristors and new hardware architectures such as 3D and self-assembly can continue the scalability of electronic systems. Neurobiology shows the way to a self-assembled existence proof of scalable, low-power computation. Investment to create the technical breakthroughs in silicon synapses, self-assembly, and scalable temporal learning algorithms can return large dividends in solving real-world problems in multi-modal surveillance and robotics.

There have been 25 years of effort in electronic implementations of neuromorphic systems. These systems have included analog and digital systems, with many being a hybrid, mixed-signal approach. Virtually all of these efforts were implemented in CMOS very large-scale integration (VLSI). The work has become increasingly sophisticated in recent years, with a variety of sensors, interconnect methods, and circuit types for neurons and synapses [9, 48, 59, 62, 63, 80, 92, 93, 97, 107, 110].

Some semi-standard methods have evolved that many in the neuromorphic engineering community use. For example, there are silicon retinas that respond only to spatial or temporal variation across pixels [78, 83]. There are silicon cochleas. There is a digital spike communication scheme called address event representation (AER) [130] which is energy-efficient and space-efficient between CMOS chips. There are local learning rules some of which involve STDP [34]. There are semi-standardized design languages and abstraction methods such as the Neural Engineering Framework (NEF). The European Union (EU) has supported much of this work, which is continuing with two large projects as part of the Human Brain Project (HBP). There are stochastic computation methods and event-driven asynchronous approaches to computation.

New devices that seem appropriate to neuromorphic CMOS are being actively investigated, including memristors, phase-change memory, and spin-torque devices [12, 74, 85, 96, 113, 111] for compact and simple-to-update synapses. There are many more synapses than any other component in neuro-inspired designs, so compact implementations are required. Furthermore, these synapses are fairly complex if they implement learning algorithms.

For electronic implementation, because the fanout of connections from each neuron is high, generally one takes advantage of multiplexing on a metal wire, using a protocol like AER, to create many connections on a single wire. In the brain, these connections are not multiplexed, so that most of the volume of the brain consists of connections. On a 2D microchip, the problem would be even worse. In addition, it is important on chip to have compact weighted connections (because there are so many) that must retain the results of learning in some form of memory. This is why there is intense interest in creating an analog, resistive, compact connection using memristors that can adjust their conductance according to the signals connecting to them locally. Another popular method for synapses involves using an analog charge that is stored on a floating gate [60].

Neuromorphic analog computing has demonstrated that for some algorithms and tasks, it is faster than digital computing and generally much lower in power dissipation. Why then, after 25 years of effort, is it still in the province of toy problems and has not had much impact on practical problems in robotics and surveillance? Partly, the answer lies with the success of the digital semiconductor industry over the same time period driven by the exponential increase in capability guided by Moore's law. Intel invested in floating gate synapses for hardware neural networks but saw greater opportunity in scaling digital CMOS and ended the project. Now that we can see the end of Moore's law on the horizon, we have an opportunity to take another look at this alternate computing style.

What is missing in serious neuromorphic computing is a robust and compact synapse technology that can be well integrated with CMOS. The TRL of analog CMOS is extremely high but not so the additions necessary to make it suitable for neuromorphic implementations, which are in the research stage. So far, memristors have shown promise but the technology is not yet ready to replace semiconductor memory, although Samsung, Hewlett-Packard, and Micron have serious efforts in phase change memory and Micron is offering a memory product. Perhaps further development will create a reliable synapse technology that can work with an appropriate, hardware-friendly learning algorithm, especially if a commercial path to a product exists.

Another strong need is a hardware-aware, temporal learning algorithm that can work in large hierarchical networks to create compositional representations. A version of STDP that works on a compact hardware synapse and that could create compositional representations of temporal events and hierarchical sequences would be a major breakthrough. A demonstration of usefulness for a large scale surveillance problem might be needed to show success. Metrics should include compact size and low power, say a thousand-fold better than digital CMOS. The European CAVIAR project [109] is an ambitious example of a vision system that did not quite show usefulness but has many of the right pieces. Funding an application-driven effort of this sort may bring all the necessary pieces together.

A promising approach might be to allow machine learning research in neuromorphic algorithms, architectures, and representations to guide hardware development. A commitment to digital simulations of neuromorphic hardware before development represents a low-cost approach that can be part of an investment strategy. The neuromorphic engineering community is beginning to pay more attention to the progress in the machine learning community, especially the success of the neural-network-driven deep learning research.

5.4 Biomolecular

In 1994 Leonard Adleman introduced biomolecular computing to solve combinatorial problems using molecules of DNA [7]. Bimolecular computing[9] remains an active area of research and, over time, has proceeded down three distinct areas of research: DNA computing, Molecular-tile computing, and Molecular computing. The goal of DNA computing and Molecular-tile computing is conventional Turing-machine computing executed using DNA molecules, though ultimately Molecular-tile computing may have more applicability as a nanotechnology. Both of these lines of investing sought to exploit the high degree of parallel execution afforded by molecular interactions in solution. Molecular computing is focused on the control of naturally occurring and synthetic biological systems and uses mostly elements of digital and more recently analog operations for its computation needs.

5.4.1 DNA-computing

DNA-computing encodes information using oligonucleotides and exploits the ligation and Watson-Crick pairing of nucleotide sequences for processing. The first DNA-based calculation was finding a Hamiltonian path for the famous *Seven Bridges of Konigsberg* problem. The calculation required nearly 7 days of lab work [7].

Computation power, in general, is a combination of the number of steps executed in parallel and the rate of execution (number of steps per unit time). This style of computing leverages the high number of concurrent molecular interactions. One gram of DNA material contains about 10^{21} base pairs. A basic operation (after the information has been encoded into the oligonucleotides) takes hours, with estimates of 10 operations per day and up to 100 operations on the optimistic side [71]

This high degree of concurrent interaction motivated the search for molecular algorithms that could be useful for solving NP-hard problems. It was known that even with this degree of concurrency, exhaustive search of the solution space that

[9] This section is restricted to computation using organic molecules (ones that contain carbon) and are commonly found in living systems. Computation using inorganic molecules is also possible and has been investigated, though not nearly as thoroughly as computing with organic molecules. This restriction was necessary because of schedule. The topic of computation with inorganic molecules merits a close examination for completeness.

scales exponentially was not practical. For example, Boolean satisfiability is an NP-complete problem with exponential computation time on a conventional computer. DNA algorithms for Boolean satisfiability of expressions with M clauses and N variables need M processing steps using 2^N strands of DNA. DNA computing trades the exponential complexity present in the number of steps in conventional computing for an exponential number of resources in the form of strands of DNA and a linear number of steps with problem size. This solution is not practical. Using all of the material in the universe would be required to solve a problem having ≈ 250 variables via this method. Such is the case for other DNA algorithms for NP-hard problems.

In 1995, JASON provided an example of using DNA computing to break 56-bit Data Encryption Standard (DES) using Lipton's algorithm [22, 76], which required 100 processing steps and 4 months of reaction time [71]. They concluded that by DNA computing did not afford any practical advantage for breaking DES over building conventional custom hardware, with the possible exception of a reduction in development time for the DNA solution given a general-purpose DNA capability. Furthermore, as they pointed out, their analysis did not account for error in the DNA computation.

5.4.2 Molecular-Tile Computing

One fundamental limitation of DNA computing is the one dimensional (1D) nature of ligation and annealing of an oligonucleotide to its complementary oligonucleotide. This one dimensional aspect limits the computation to proceed by growing longer and longer DNA molecules. While there were a number of proposals to extend DNA computing into a universal Turing Machine [14, 101, 117], none appeared practical, which is also likely why they were never realized. An alternative was the introduction of *Molecular-tile* computing that allows for the calculation to proceed in two (or more) spatial dimensions. Double-crossover (DX) molecules consist of two side-by-side double-stranded helices linked at two crossover junctions, giving them a tile-like nature with edges composed of single strands of DNA [47, 73]. These tiles can stick to edges of other tiles that have the complementary DNA sequences. Proceeding in this way, 2D structures can be grown that are calculations.

Molecular tiling is Turing complete and, hence, is capable of the same computation as conventional computing. Ideal tile calculations (for which the bonding of two tiles is perfect) are first described by Wang in 1960 [123] and are later shown to be Turing complete [100]. Rothemund and Winfree extended this theory by including a parameter describing the cooperativity of tiles to bind. They also establishes that molecular tiling is Turing complete provided the binding parameter exceeds a threshold [103].

Winfree was the first to demonstrate tile computing in 1998 [125, 128]. This demonstration used only two tile types and was more a demonstration of programmed self-assembly rather than a calculation. Subsequent demonstrations have shown cascaded exclusive OR (XOR) operations and counting to 17 with several errors [13].

Like any crystal growth technique, molecular-tile computation is governed by thermodynamics. Winfree analyzed the thermodynamics of DNA self-assembly through DNA hybridization and determined that the lowest error rate occurs at the melting temperature when crystal growth is slowest and that error rates can be made arbitrarily low by decreasing concentration and increasing binding strengths [127]. In other words, the rate at which tiles attach is only slightly greater than the rate at which they release. Large molecular-tile calculations require low concentrations and, hence, computation proceeds very slowly. There has been some thought on the thermodynamic is truly a fundamental limitation and speculation on ways to get around it, including local parallelism [98] and use of an additional energy source for biological proofreading [69]. No experimental evidence supports these approaches for molecular-tile computing.

Algorithms for molecular-tile computations have also been explored. The best nondeterministic molecular-tiling algorithms for Boolean satisfiability [27, 28] of expressions with M clauses and N variables are linear in time with M and use 64 tile types. The probability of success per attempt is at least $\left(\frac{1}{2}\right)^N$. Hence, a calculation using an exponential number of seeds, 2^N, has a 63% probability of success. Similar results hold for factoring integers [26] using Molecular-tile computing. Factoring an N-bit number can use a constant number of tile types with a probability of success of $\left(\frac{1}{6}\right)^N$. Similarly, using 6^N seeds has a 63% chance of correctly factoring the number.

These results should not be surprising. Molecular-tile computation is still a Turing-machine model of computation, and, consequently, the overall complexity of the problem remains exponential. Rather than running a single calculation with exponential running time on a conventional computer, Molecular-tile computing (and DNA computing) simultaneously runs an exponential number of linear-time calculations. The overall amount of work is still exponential. While these two forms of bimolecular computation are able to achieve an unprecedented amount of parallel computation compared to conventional computing, the exponential growth of molecules needed for the computation still limits biomolecular to rather modest-sized problems.

More recent work in this area has shifted from self-assembly (of molecular tiles) for computation and towards understanding self-assembly of 2D and 3D structures [39, 41, 102]. This ability to engineer higher dimensional structures and combine them with elementary biomolecular logic operations is finding novel application in Molecular computing systems [40]. Douglas et al. fabricated an autonomous DNA nanorobot capable of transporting molecular payloads selectively to biological cells. The selection mechanism used an aptamer-encoded logic gate to sense cell surfaces and deliver the payload to the cell [40]. This application of engineered nanostructures and molecular computing may represent an important future drug delivery mechanism.

5.4.3 Molecular Computing

Both DNA computing and Molecular-tile computing were primarily addressing hard computational problems. *Molecular computing* is developing approaches to incorporate elemental processing into biological systems. The Molecular computing perspective incorporates elements of systems biology, chemical reaction networks, and control theory to provide molecular-based solutions for information-processing tasks in naturally occurring and synthetic biological systems. It is not primarily driven by computation scale or speed. Rather, it is focused on the need to work in a biological environment. Driving applications for Molecular computing are in medicine, energy production, and environmental engineering.

Molecular computing has mostly been focused on elementary logic operations and limited efforts with state machines. It is still in its infancy. Recently, there has been increased interest in analog computing, primarily as a way of coping with limited computational resources in biological systems. Viewing molecular computing in isolation as a computing technology is misleading. Rather, it should properly be viewed as an elemental technology in a complex system. This view will drive its development. However, it may also be a disruptive technology and find applications not originally envisioned.

The range of work in this area is vast and cannot be completely summarized in this report; however, we will give a few select examples of the technology. There are excellent recent reviews of the technology [17, 86] and some of the broader issues [29, 31, 43, 67, 68]. Molecular logic computing devices have used DNA, RNA and protein molecules.[10] Time scales for logic operation range from 10 seconds to days for *in vivo* and range from hours to days for *in vitro* depending on the mechanism[11]. DNA-based gates have operation times on the order of hours, which is the same time scale for DNA-computing described earlier. In that case, the computational power stems from the degree of concurrency and not from the speed of interaction. All three molecules have been shown to work *in vitro* and *in vivo*. Unlike transistors, which stay put on the chip, molecules in solution move, and, consequently, this causes many unwanted interactions that limit scalability *in vitro*. One important feature of *in vivo* technology is the cell boundaries that limit unwanted interactions. Another challenge has been building networks of logic devices, which requires the output of one logic element to serve as the input of another element. Many of the demonstrations are inherently not cascadable, though some, including DNA logic circuits and protein-based enzymes, have demonstrated cascaded operation [88, 94, 95]. DNA interactions can be readily interpretable in terms of conventional logic, whereas many of the other mechanisms used for logic do not allow for this interpretation. For example, consider elementary logic performed with RNA devices [125]. These devices process molecular input to targeted protein outputs. The authors demonstrated operation of an AND gate that only responded with high production of green fluorescent protein (GFP)

[10] Table 1 [86] provides a concise summary of these demonstrations as of late 2012.

[11] See Figure 2 [86] for the time scales of the various mechanisms.

when the level of theophyline and tetracycline were sufficiently high.

Demonstrations have also included state machines. Ultimately, this direction can lead to low-energy computation as discussed by Bennett [20] and can be realized, in principle, with RNA molecules. Bennett refers to this as *Brownian computing.* Another feature of this low-energy computation is that it is only low energy when the computation proceeds slowly. Hence, computational power in this situation stems from performing many operations in parallel with a minimal number of sequential steps. This theme is common and recurrs often in biomolecular computing.

In 2003, Stojanovic and Stefanovic demonstrated a DNA-based molecular automaton capable of interactively playing tic-tac-toe against a human opponent [121]. It incorporated 23 molecular-state logic gates and optically active DNA arrayed in nine wells (3 × 3) array. In 2004, Benenson et al. demonstrate a molecular computer for logical gene control [18]. The system operated *in vitro* to sense and analyze the levels of messenger RNA and, in response, produced a molecule capable of affecting the level of gene expression. State machines have also been constructed *in vivo* [11].

More recently, molecular computation has also been used for analog computing. The motivation for this stems in part from the need to do efficient computation in the resource-limited environment of cells. Daniel et al. demonstrated analog computation in living cells using just three transcription factors [35]. They demonstrated an analog adder circuit that summed the molarity of two different protein inputs. This demonstration is a first step towards multi-signal integration and processing.

5.4.4 Neuromorphic Computing

Compared to the proliferation of CMOS designs, there have not been many attempts at using bio-molecular methods to implement neuro-inspired computational designs. Given the recent dramatically increasing accumulation of knowledge in neuroscience and bio-molecular methods, new computational techniques may become available in the near future. In particular, more experimental details on STDP [34], and learning in biological neurons and synapses may inform computational methods of how to learn temporal patterns in a hierarchical structure that has semantic significance.

The burgeoning field of synthetic biology has spawned convincing recent demonstrations of analog and digital computation [33, 42, 46, 49, 51, 57, 87, 99, 126] that have been engineered using synthetic biology methods. These methods can lead to computers, that operate on biochemical and optical inputs. The computers will be much slower than electronic computers, but can compute with very large-scale parallelism on a molecular or cellular scale. For general computation, a method of interfacing with existing data sources is needed. Optogenetic [23, 75] methods for neuromodulation in nerve cells may be adapted for such purpose. However, it seems unlikely that sufficient data bandwidth would be available to fully exploit the computational parallelism of bio-molecular computing.

All of the biomolecular technology is at the lowest TRL. Basic principles have been demonstrated and reported. The community regarding the technology concept and application area. DNA computing has the potential to rapidly advance; however, it lacks a driving application. Because DNA computing is nearly 20 years' old, this situation is not likely to change. Molecular-tile computing as a means of calculation (as opposed to a technology for self-assembly) shows more potential, but no clear case has been presented that shows its advantage. This area needs to be further explored. Molecular computing is still in its infancy. Its potential is high and will ultimately be driven by the industrial bases it serves: medicine, energy production, and environmental engineering.

At present, all of this technology is prone to errors and is not robust. Polymerase chain reaction (PCR) amplifies DNA and has an error rate of one base pair in 10^5. If PCR is a significant part of the calculation (rather than just a read-out technology), then the errors caused by PCR can be critical and need to be addressed. Many of the Molecular computing logic gates depend on chemical concentrations, and verifying that a biological computation circuit computes according to specification under all possible conditions is challenging, and even more so for *in vivo*.[12]

5.5 Quantum Technologies

Many candidate material systems have been proposed, and their properties have been explored as candidates for building a quantum computer. Some are better than others, of course, and some are not even acceptable. The three most viable material systems—trapped-ion, superconducting, and solid-state—are discussed in this section. Of these, trapped-ion is by far the most mature, though questions remain regarding its scalability. Although not as mature as trapped-ion, approaches based on superconductivity are maturing, and have demonstrated multiple gate operations many times. Of the three, solid-state is the least mature but ultimately offers the best chance for scalability. Solid-state has been a catch-all for many different approaches based on condensed-matter material systems. The discussion here is restricted to electrically addressed quantum dots in gallium arsinide and silicon and dopants in silicon. These approaches were selected because of their ability to be controlled electrically. The most notable omissions from this list are optical quantum dots and vacancies in diamond, both of which would be controlled via optical signals rather than electrical signals.

These choices were made for practical reasons. Optical beams present challenges in addressing and controlling a large number of qubits. Although optics may be needed for connecting multiple quantum processors at some point in the future, the most sensible approach at present is to place as much capability into a single processor before considerin scaling through sharing of quantum information between many discrete processors. The optical requirements are also present for trapped-ions (though some schemes hold out the possibility of using microwaves) and add significant

[12] See the **Methodological Section [17] for additional details**

complications to the system. A guiding principle in making these choices has been to examine the supporting technology necessary to realize a quantum computer. In this case, supporting technology includes all of the ancillary technology, such as packaging and control that will be necessary to realize a quantum computer. Reasoning along these lines leads one to conclude that, as valuable as wires and electrical control have been to transistors, they will also be as valuable and economical for quantum technology in the long run. Hence, those technologies that are best able to leverage off the silicon VLSI industry stand the best chance of producing a capability quantum computer—a quantum computer capable of significantly outperforming the largest practical classical computer.

Although some technology may encounter practical limits, it may still be useful for capability quantum computers. For example, a quantum computer for quantum simulation may only need on order of 100 qubits to be of computational value, whereas quantum computers capable of factoring 1,000-bit numbers may need 100,000 to 100,000,000 qubits, depending on the resource needs for fault tolerance. Hence, material systems that are limited in scalability may still be of practical value for certain classes of problems, whereas other problems will need a technology that is intrinsically more scalable. Hence, a technology that may not be inherently scalable, such as trapped-ion, may nevertheless be valuable for other applications while also being useful as a stepping stone towards quantum computers of incrementally higher capability.

Interest in quantum computing took off in 1994 with the discovery of Shor's algorithm for factoring numbers. In the intervening 20 years, while much progress was made, a logical qubit has yet to be demonstrated. A logical qubit is one that can retain its quantum state for the duration of the calculation. Recently the the first demonstration of extending the coherence lifetime through quantum error techniques has been demonstrated; however, there is still a long way to go. It is important to keep in mind that the challenges faced by each of the three material systems are quite different. Nature presented trapped-ions with a high-precision qubit. This is not the case with either superconducting or solid state. As a consequence, solid-state is still trying to build a reliable qubit, whereas trapped-ion could largely skip this step (though it was necessary to learn how to hold and manipulate the ions). Superconductivity is positioned between the two. Qubits are defined lithographically and fabricated using approaches similar to those used in electronics. Each of the three approaches has its own advantages and disadvantages. Furthermore, because we are still at such an early stage, the most informative strategy is to continue development on all three approaches. One benefit of this strategy is that trapped-ion, for instance is experimentally investigating interaction among multiple qubits that may likely impact the architecture and design of other approaches. For example, research at The University of Innsbruck has observed that the practical effects of correlated noise have a significant impact on decreasing the coherence lifetime of the qubit [5].

6 Potential Follow-On Work: Deep Dives

At present, there is no alternative to the silicon device platform offers a clear and compelling computational advantage across a broad spectrum of applications. Although the silicon device platform will continue to evolve over the next several decades driven by industry's $50 billion per year investment, the direction will be focused on the needs of large consumer markets and will likely not be optimal for many IC applications. In this chapter, we offer a vision of computing based on the silicon device platform and outline an investment strategy to expand computational capabilities of core interest to the IC and to accelerate the development of select computational capabilities of common interest. We assume that silicon can comprehensively provide for classical digital, neuro-inspired, and analog computational styles. It may even be the best choice for digital quantum computation, though that idea will not be explored further in this chapter. The ideas present in this chapter should be regarded as a starting point and merit a more detailed exploration before committing to a way forward.

The overarching premise of this vision is that fully leveraging the market-driven investment of the commercial sector in the silicon platform is not only low risk, but also offers the best return on investment. Consequently, the focus of strategy is on leveraging the silicon device platform. Smart investment will provide innovation over the next two decades. On a longer time-scale, synthetic biology may offer a disruptive computational platform for select applications and should be watched. Commercial investment for synthetic biology will come mostly from medicine, energy production, and environmental engineering applications. It is unlikely that synthetic biology will provide a computation platform as versatile as silicon; nevertheless, it may offer significant advantages for neuro-inspired computing. These advantages may extend to analog computing but it is unlikely that high-performance digital computing will realize any advantage.

This investment strategy is developed around the the styles of classical digital, neuro-inspired, and analog computing, in that order. Of the three, classical digital computing will receive the most attention from industry, and making progress in this area will require need commercial relationships with industry leaders. The other two styles, neuro-inspired and analog, receive much less attention from industry and consequently present different challenges. The challenge for neuro-inspired stems from needing a strong focus at the device level, guided by synergistic interaction across algorithms, neuroscience, and engineering. From an IC perspective, analog computing is challenged by a lack of applications in the IC. In this study, we uncovered approaches for mapping hard discrete math problems (Boolean satisfiability) onto dynamical systems, with the potential of solving these systems using analog computation. The ultimate goal is to develop mixed computation style devices (i.e., devices capable of supporting Boolean computation, neuro-inspired computation, analog computation, and possibly some day, quantum computing). The approach is to tactically focus on each style separately at the device level while strategically focusing the goal of mixed-style devices.

6.1 Classical Digital Computation

The following topics merit further exploration towards expanding future capabilities of classical digital computation:

- New memory technology for static random-access memory (SRAM).
- Scale CMOS to its practical limit,
- Device technology alternative to CMOS,
- Extend CMOS to a heterogenous integration platform,
- Develop new functional computational cores, and
- Theory and practice of PIM computing.

The first four of these focus area are listed in the ITRS roadmap as long-term challenges [3] in the emerging research devices and materials section. An approach coordinated with industry will be needed. The last two focus areas are focused towards providing new computational capabilities at the functional and system level.

6.1.1 New Memory Technology for SRAM

SRAM memory cells are large, leading to low integration density, and can be a significant portion of the chip power budget. DRAM is significantly lower power but cannot be embedded on processor chips. Industry focus is on replacing SRAM and possibly FLASH memories by 2018. Embeddable memory provides for architectural options, such as PIM and other functional cores. Because industry is facing significant limitations with current approaches, it is aggressively pursuing alternatives, and investment by the IC is not recommended at present. Nevertheless, the IC will benefit by closely following progress.

6.1.2 Scale CMOS to its Practical Limit

This is a core challenge for industry and consequently requires no investment from the IC. Industry predicts to reach this goal in the 2023 timeframe. Furthermore, the projections are that the gate length will be only 3x larger than the fundamental limit of scaling an electronic charge-based switch and that the scaling is practically limited by the maximum allowable power dissipation of $\approx$100 W/cm^2 [131]. Even if unforeseen challenges arise that prematurely limit scaling of CMOS, there is no practical investment from the government that will break through the barrier, given the singular importance scaling CMOS has for industry.

6.1.3 Device Technology Alternative to CMOS

This challenge addresses alternatives to CMOS after it reaches its practical limit around 2023. The primary motivation for replacing CMOS is to continue scaling - increase device density and lower electrical power. Several computational state variables are being considered. The largest class is charge-based, many of which are incorporating novel materials, such as graphene [108]. This area is already well-covered, and the best choice of device should be left to industry.

Encoding information as a difference in the quantity of charge necessarily leads to energy dissipation when changing state. Alternative state variables, including electron spin [15], magnetic domains [89] and others offer the possibility of lower energy dissipation in switching and transport, depending on distance. Specifically, the areas of spin-wave, nonmagnetic, and all-spin logic are within the scope of the industry, but they are considered long-term options. The government could accelerate this research. This should lead to lower power cores for fundamental operations such as floating point units.

State variables other than charge are likely not good candidates for long interconnects on chips. Attenuation coefficients are large and lead to exponential attenuation of signals with distance. The attenuation coefficient for spin waves is a millimeter and microns for plasmons [32]. The attenuation coefficient of metal interconnects is on the order of a centimeter and consequently most of the power dissipation results from discharging the capacitance of the wire, which is linear with wire length. Consequently, charge base interconnects are likely to remain the best choice in the foreseeable future for long interconnects. Carbon nanotubes have been demonstrated to have significantly lower electrical resistance and a capacitance similar to that of copper interconnects [72, 118]. The lower resistance of single-walled CNTs results in fewer repeaters (necessary to mitigate dispersion effects) on long interconnects and this can result in up to an order of magnitude reduction in dissipation. Hence, CNTs may significantly improve interconnect performance for long interconnects. The cost to develop CNT interconnects is likely prohibitive for the government without significant support from industry. Another interesting possibility are topological insulators. These material offer the possibility of near perfect electrical conductivity at room temperature [129]

6.1.4 Extend CMOS to a Heterogenous Integration Platform

This is also a challenge called out in the ITRS roadmap, but unlike the ones discussed above, the government should proactively influence the capability of the integration platform. At present the ITRS roadmap does call for it to support "new device technologies and primitive-level architecture to provide special purpose optimized functional cores heterogeneously integrable with CMOS." For the ultimate goal of mixed-style computation device to be realized the platform must support neuro-inspired computational cores and analog cores. The architecture should have a strong emphasis on data transport and organization of the computation. The architecture should extend beyond the computational accelerator and provide support for unconventional, reconfigurable, and programmable computation.

6.1.5 Develop New Functional Cores

Applications for which a key requirement is reduced power consumption will increasingly benefit from specialized processing tuned to the application. Consequently, the ability to include special-purpose functional cores at the device level will be advantageous. The effort is closely related to using CMOS as a heterogeneous integration platform. We expect that a major focus will be the integration of processing and memory in these cores, possibly in the from of PIM devices and associative memories.

The success and outcome of developing new functional cores and extending CMOS to a heterogenous integration platform will depend critically on choosing good computational problems. We do not have a strong opinion on which problems in the IC provide the best focus for these innovations. One possibility is Boolean satisfiability, though this question merits significant deliberation as part of the investment strategy.

6.1.6 Theory and Practice of Processor-in-Memory Computing

Because most of the energy budget of computation on modern digital architectures is devoted to moving data between processor and memory, industry may be moving towards architectures in which additional processing elements are collocated with memory. Architectures in which memories have processing capability are known generally as PIM computers. (The RAM in such machines has also historically been called “intelligent RAM” or “computational RAM.”) Due to the growing chasm between CPU performance and memory performance, PIM architectures are perhaps our best hope for sustained increases in computational capability from CMOS hardware. However, it is not clear how best to program such non-von Neumann systems.

There are signs that the hardware industry is already moving in the direction of collocating processing with memory. The Hybrid Memory Cube (HMC), in which memory modules are packaged in three dimensions to increase density and reduce latency, includes a logic layer. The logic layer hosts the memory controller but has remaining space for additional computational elements. Most of the top performing machines in the Top 500 make use of hardware accelerators such as Graphic Processing Units (GPUs) or the Xeon Phi, both of which can be thought of as random access memory with collocated processors that have high-bandwidth, low-latency access to local memory.

PIM computing was a popular research topic in the late 1990s and early 2000s, but sees less attention today [see, for example, 91]. The nature of the work done on PIM consists primarily of performance and cost estimates, with some proof-of-concept hardware being built [53, 91]. While some progress was made in our understanding of how to program a PIM architecture, this progress is the result of considering a handful of applications on the architecture, and it is not clear that an underlying theory of PIM programming developed [25, 56, 120]. (It should be noted, however, that the IC may have considerable experience with PIM architectures, as evidenced by the Terasys [53]. Before embarking on the PIM research program described below, the IC's history and experience with PIM should first be investigated.)

We envision a two-pronged, two-phase effort whose goal is to maximize our ability to program increasingly heterogeneous computers that are approaching a PIM architecture. In phase one, one prong of the effort will develop a theoretical understanding of how to program PIM computers, including tools for performance prediction as a function of system design parameters. This understanding is analogous to the detailed understanding we have of von Neumann machines, in which data moving to and from memory is the paradigm. The other prong in phase one of the effort will port a range of applications (of interest to the IC, but perhaps more broadly) to PIM architectures. The applications could be drawn from the interprocess communication families ("dwarves") identified in Asanovic et al. [10]. With the theoretical understanding and practical experience gained from phase one, domain-specific compilers and languages will be designed and built in phase two. The domain-specific compilers will support some general-purpose, object-oriented language [object-oriented languages are a very natural fit for PIM; see, for example, 25]. The domain-specific languages will support those domains that are deemed most promising to benefit from PIM and most relevant for the IC. These domains might be, for example, discrete mathematics and graph analysis. As pointed out in JASON [65], domain-specific compilers and languages may accelerate code development in an era of rapidly evolving hardware.

6.2 Neuro-Inspired Computation

The following topics merit further exploration towards expanding future capabilities of neuro-inspired computation:

- Silicon-compatible devices for neuron, synapse, axon, and dendrite, and
- Interdisciplinary understanding of neural structure, computational model, and algorithms.

Neuro-inspired computation is significantly less mature than classical digital or even analog computation. While much focus has been on learning mechanisms, such as STPD, the overall computational model and means merits more holistic consideration over the three areas mentioned previously.

6.2.1 Silicon-Compatible Devices for Neuron, Axon, Synapse, and Dendrite

It is really the synapse that is the most challenging. The synapse allows an amount of current to pass from the axon of one neuron to the dendrite of another neuron. Learning adjusts the amount of current that flows. Memristors are leading candidate for silicon synapses. They are two-terminal devices. An alternative is the synaptic transistor, which is a three terminal device. Synaptic transistors are traditionally realized using floating gate transistors and more recently using adaptive oxides [55, 114, 132], which are in the same family as the materials used for memristors. Properties of adaptive oxides, such as resistivity, can be modified electrically in a non-volatile manner. The goal of government investment would be to develop a synaptic transistor and understand its advantages from an architectural perspective.

6.2.2 Interdisciplinary Understanding of Neural Structure, Computational Model, and Algorithms

There is still much that we do not understand about how the brain works. Furthermore the performance of our best artificial intelligence (AI) algorithms is still lacking compared to the capability of the brain. We need a serious interdisciplinary effort to explore these issues and guide our development of this style of computation. This topic has been a consistent theme throughout this paper.

6.3 Analog Computation

The following topics merit further exploration towards expanding future capabilities of analog computation:

- Role of analog computation, and
- Interaction of analog computation with other computation styles.

Analog computing is relatively mature, though it has fallen out of favor because of advances in digital computing and the promise of neuro-inspired computing.

6.3.1 Role of Analog Computing

One possibility is the mapping of problems into dynamical systems and solving for the evaluation using an analog computation. Such an approach was recently developed for Boolean satisfiabilty.

6.3.2 Interaction of Analog Computation with Other Computation Styles

CMOS is a sound platform for analog computing, though we need to understand its role in computation for the IC and its interaction for support with other computation styles. The ultimate computation potential may lie in the way various styles are mixed in the solution.

7 Abbreviations

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
AC	alternating current
AER	Address-Event Representation
AI	artificial intelligence
ASIC	application-specific integrated circuit
C3	Cyrogenic Computing Complexity
CMOS	complementary metal-oxide semiconductor
CPU	central processing unit
DARPA	Defense Advanced Research Projects Agency
DC	direct current
DES	Data Encryption Standard
DNA	deoxyribonucleic acid
DOE	Department of Energy
DRAM	dynamic random-access memory
DX	double-crossover
EAC	extended analog computer
EU	European Union
FET	field-effect transistor
FLOPS	floating point operation per second
GFLOPS	giga (10^9) floating point operations per second
GIOPS	giga (10^9) instruction operations per second
GFP	green fluorescent protein
GPAC	general-purpose analog computer
GPU	graphics processing unit
HBP	Human Brain Project
HMC	human memory cube
IARPA	Intelligence Advanced Research Projects Activity
IC	Intelligence Community
IOPS	instruction operations per second
ITRS	International Technology Roadmap for Semiconductors
LIGO	Laser Interferometer Gravitational-Wave Observatory
MEMS	microelectromechanical systems
MRI	Magnetic Resonance Imaging
NEF	Neural Engineering Framework
NMOS	N-type metal-oxide semiconductor
NSA	National Security Agency

ODE	ordinary differential equation
PCR	polymerase chain reaction
PIM	processor in memory
PFLOPS	peta (10^{15}) floating point operations per second
PIOPS	peta (10^{15}) instruction operations per second
PUM	processor under memory
QTM	quantum Turing machine
RAM	random-access memory
RLC	resistance, inductance, and capacitance
RNA	ribonucleic acid
SFQ	simple flux quanta; simple flux quantum
SOI	silicon on insulator
SPD	special-purpose device
SRAM	static random-access memory
STDP	spike-timing dependent plasticity
TRL	Technology Readiness Level
USG	United States Government
VSLI	very large-scale integration
XOR	exclusive OR

8 Acknowledgments

The authors acknowledge John Fregeau for his contributions to this report, including sections 4.1, 4.3.1, and 4.5.

References

[1] "IC Insights,"
http://www.icinsights.com/data/articles/documents/443.pdf

[2] "Intel Corporation,"
http://www.intel.com/content/www/us/en/history/museum-story-of-intel-4004.html

[3] "ITRS Home,"
http://www.itrs.net/

[4] "Semiconductor Industry Association,"
http://www.semiconductors.org/news/2014/02/03/global_sales_report_2013/semiconductor_industry_posts_record_sales_in_2013/

[5] "14-qubit entanglement: Creation and coherence," Physical Review Letters, Vol. 106, p. 130506, 2011

[6] Aaronson, S., "Guest column: NP-complete problems and physical reality," ACM Sigact News, Vol. 36, pp. 30–52, 2005

[7] Adleman, L., "Molecular computation of solutions to combinatorial problems," Science, Vol. 266, pp. 1021–1024, 1994
http://www.ncbi.nlm.nih.gov/pubmed/7973651http://www.sciencemag.org/cgi/doi/10.1126/science.7973651

[8] Alaghi, A. & Hayes, J. P., "Survey of Stochastic Computing," ACM Trans. Embed. Comput. Syst., Vol. 12, pp. 92:1–92:19, 2013
http://doi.acm.org/10.1145/2465787.2465794

[9] Alspector, J., Gupta, B., & Allen, R. B., "Performance of a stochastic learning microchip," in *Artificial neural networks*, pp. 66–78, IEEE Press, 1990

[10] Asanovic, K., et al., "The landscape of parallel computing research: A view from berkeley," Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Tech. rep., 2006
https://www.classle.net/sites/default/files/text/33631/The_Landscape_of_Parallel_Computing_Research.pdf

[11] Ausländer, S., Ausländer, D., Müller, M., Wieland, M., & Fussenegger, M., "Programmable single-cell mammalian biocomputers." Nature, Vol. 487, pp. 123–7, 2012
http://www.ncbi.nlm.nih.gov/pubmed/22722847

[12] Bamford, S., Murray, A., & Willshaw, D., "Silicon synapses self-correct for both mismatch and design inhomogeneities," Electronics Letters, Vol. 48, p. 360, 2012
http://digital-library.theiet.org/content/journals/10.1049/el.2012.0257

[13] Barish, R. D., Schulman, R., Rothemund, P. W. K., & Winfree, E., "An information-bearing seed for nucleating algorithmic self-assembly." Proceedings of the National Academy of Sciences of the United States of America, Vol. 106, pp. 6054–9, 2009
http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2660060&tool=pmcentrez&rendertype=abstract

[14] Beaver, D., "Molecular Computing," Pennsylvania State University, Tech. Rep. 814, 1995

[15] Behin-Aein, B., Datta, D., Salahuddin, S., & Datta, S., "Proposal for an all-spin logic device with built-in memory," Nature Nanotechnology, Vol. 5, pp. 266–270, 2010
http://www.nature.com/nnano/journal/v5/n4/abs/nnano.2010.31.html

[16] Bekenstein, J. D., "Black holes and entropy," Physical Review D, Vol. 7, p. 2333, 1973

[17] Benenson, Y., "Biomolecular computing systems: principles, progress and potential." Nature reviews. Genetics, Vol. 13, pp. 455–68, 2012
http://www.ncbi.nlm.nih.gov/pubmed/22688678

[18] Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., & Shapiro, E., "An autonomous molecular computer for logical control of gene expression." Nature, Vol. 429, pp. 423–9, 2004
http://www.ncbi.nlm.nih.gov/pubmed/15116117

[19] Bennett, C. H., "Logical reversibility of computation," IBM journal of Research and Development, Vol. 17, pp. 525–532, 1973

[20] Bennett, C. H., "The thermodynamics of computation—a review," International Journal of Theoretical Physics, Vol. 21, pp. 905–940, 1982
http://link.springer.com/10.1007/BF02084158

[21] Blum, L., Shub, M., & Smale, S., "On a theory of computation and complexity over the real numbers: NO-completeness, recursive functions and universal machines," Bulletin of the American Mathematical Society, Vol. 21, 1989

[22] Boneh, D., Dunworth, C., Lipton, R. J., & Sgall, J., "On the computational power of DNA," Discrete Applied Mathematics, Vol. 71, pp. 79–94, 1996
http://linkinghub.elsevier.com/retrieve/pii/S0166218X96000583

[23] Boyden, E. S., Zhang, F., Bamberg, E., Nagel, G., & Deisseroth, K., "Millisecond-timescale, genetically targeted optical control of neural activity," Nature neuroscience, Vol. 8, pp. 1263–1268, 2005

[24] Branicky, M. S., "Analog computation with continuous ODEs," in *Physics and Computation, 1994. PhysComp'94, Proceedings., Workshop on*, pp. 265–274, IEEE, 1994

[25] Brockman, J. B., Kogge, P. M., Sterling, T. L., Freeh, V. W., & Kuntz, S. K., "Microservers: A new memory semantics for massively parallel computing," in *Proceedings of the 13th international conference on Supercomputing*, p. 454463, 1999
http://dl.acm.org/citation.cfm?id=305234

[26] Brun, Y., "Arithmetic computation in the tile assembly model: Addition and multiplication," Theoretical Computer Science, Vol. 378, pp. 17–31, 2007
http://linkinghub.elsevier.com/retrieve/pii/S0304397506007894

[27] Brun, Y., "Nondeterministic polynomial time factoring in the tile assembly model," Theoretical Computer Science, Vol. 395, pp. 3–23, 2008
http://linkinghub.elsevier.com/retrieve/pii/S0304397507006287

[28] Brun, Y., "Improving Efficiency of 3-SAT-Solving Tile Systems," pp. 1–12, 2011

[29] Bunka, D. H. J. & Stockley, P. G., "Aptamers come of age - at last." Nature reviews. Microbiology, Vol. 4, pp. 588–96, 2006
http://www.ncbi.nlm.nih.gov/pubmed/16845429

[30] Chang, F., et al., "Bigtable: A Distributed Storage System for Structured Data," ACM Transactions on Computer Systems, Vol. 26, pp. 1–26, 2008
http://portal.acm.org/citation.cfm?doid=1327452.1327492http://portal.acm.org/citation.cfm?doid=1365815.1365816

[31] Chen, Y. Y., Galloway, K. E., & Smolke, C. D., "Synthetic biology: advancing biological frontiers by building synthetic systems." Genome biology, Vol. 13, p. 240, 2012
http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3334564&tool=pmcentrez&rendertype=abstract

[32] Conway, J. A., Sahni, S., & Szkopek, T., "Plasmonic interconnects versus conventional interconnects: a comparison of latency, crosstalk and energy costs," Optics Express, Vol. 15, pp. 4474–4484, 2007
http://www.opticsexpress.org/abstract.cfm?URI=oe-15-8-4474

[33] Culler, S. J., Hoff, K. G., & Smolke, C. D., "Reprogramming Cellular Behavior with RNA Controllers Responsive to Endogenous Proteins," Science, Vol. 330,

pp. 1251–1255, 2010, PMID: 21109673
http://www.sciencemag.org/content/330/6008/1251

[34] Dan, Y. & Poo, M.-m., "Spike Timing-Dependent Plasticity of Neural Circuits," Neuron, Vol. 44, pp. 23–30, 2004
http://www.sciencedirect.com/science/article/pii/S0896627304005768

[35] Daniel, R., Rubens, J. R., Sarpeshkar, R., & Lu, T. K., "Synthetic analog computation in living cells." Nature, Vol. 497, pp. 619–23, 2013
http://www.ncbi.nlm.nih.gov/pubmed/23676681

[36] Dean, J. & Ghemawat, S., "MapReduce: Simplified Data Processing on Large Clusters," Communications of the ACM, Vol. 51, p. 107, 2008
http://portal.acm.org/citation.cfm?doid=1327452.1327492

[37] Deutsch, D., "Quantum theory, the Church-Turing principle and the universal quantum computer," Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, Vol. 400, p. 97117, 1985
http://rspa.royalsocietypublishing.org/content/400/1818/97.short

[38] Dlugosch, P., Brown, D., Glendenning, P., Leventhal, M., & Noyes, H., "An Efficient and Scalable Semiconductor Architecture for Parallel Automata Processing," to appear in IEEE Transactions on Parallel and Distributed Systems, 2013

[39] Doty, D., "Theory of algorithmic self-assembly," Communications of the ACM, Vol. 55, p. 78, 2012
http://dl.acm.org/citation.cfm?doid=2380656.2380675

[40] Douglas, S. M., Bachelet, I., & Church, G. M., "A logic-gated nanorobot for targeted transport of molecular payloads." Science, Vol. 335, pp. 831–4, 2012
http://www.ncbi.nlm.nih.gov/pubmed/22344439

[41] Douglas, S. M., Dietz, H., Liedl, T., Hogberg, B., Graf, F., & Shih, W. M., "Self-assembly of DNA into nanoscale three-dimensional shapes." Nature, Vol. 459, pp. 414–8, 2009
http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2688462&tool=pmcentrez&rendertype=abstract

[42] Elowitz, M. B. & Leibler, S., "A synthetic oscillatory network of transcriptional regulators," Nature, Vol. 403, pp. 335–338, 2000
http://www.nature.com/nature/journal/v403/n6767/abs/403335a0.html

[43] Endy, D., "Foundations for engineering biology." Nature, Vol. 438, pp. 449–53, 2005
http://www.ncbi.nlm.nih.gov/pubmed/16306983

[44] Ercsey-Ravasz, M. & Toroczkai, Z., “Optimization hardness as transient chaos in an analog approach to constraint satisfaction,” Nature Physics, Vol. 7, pp. 966–970, 2011
http://www.nature.com/nphys/journal/v7/n12/abs/nphys2105.html

[45] Feynman, R. P., “Simulating physics with computers,” International Journal of Theoretical Physics, Vol. 21, p. 467488, 1982
http://www.springerlink.com/index/T2X8115127841630.pdf

[46] Friedland, A. E., Lu, T. K., Wang, X., Shi, D., Church, G., & Collins, J. J., “Synthetic Gene Networks That Count,” Science, Vol. 324, pp. 1199–1202, 2009, PMID: 19478183
http://www.sciencemag.org/content/324/5931/1199

[47] Fu, T. J. & Seeman, N. C., “DNA double-crossover molecules.” Biochemistry, Vol. 32, pp. 3211–20, 1993
http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1302964&tool=pmcentrez&rendertype=abstract

[48] Furber, S., et al., “Overview of the SpiNNaker System Architecture,” IEEE Transactions on Computers, Vol. 62, pp. 2454–2467, 2013

[49] Gardner, T. S., Cantor, C. R., & Collins, J. J., “Construction of a genetic toggle switch in Escherichia coli,” Nature, Vol. 403, pp. 339–342, 2000
http://www.nature.com/nature/journal/v403/n6767/abs/403339a0.html

[50] Ghemawat, S., Gobioff, H., & Leung, S.-t., “The Google file system,” ACM SIGOPS Operating Systems Review, Vol. 37, p. 29, 2003
http://portal.acm.org/citation.cfm?doid=1165389.945450

[51] Gibson, D. G., et al., “Creation of a Bacterial Cell Controlled by a Chemically Synthesized Genome,” Science, Vol. 329, pp. 52–56, 2010, PMID: 20488990
http://www.sciencemag.org/content/329/5987/52

[52] Glanz, J., “Google Details Electricity Usage of Its Data Centers,” The New York Times, 2011
http://www.nytimes.com/2011/09/09/technology/google-details-and-defends-its-use-of-electricity.html

[53] Gokhale, M., Holmes, B., & Iobst, K., “Processing in memory: The Terasys massively parallel PIM array,” Computer, Vol. 28, p. 2331, 1995
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=375174

[54] Google, “Google See where the Internet lives,” , 2013
http://www.google.com/about/datacenters/gallery/#/places

[55] Ha, S. D. & Ramanathan, S., “Adaptive oxide electronics: A review,” Journal of Applied Physics, Vol. 110, 2011

[56] Hall, M., et al., “Mapping irregular applications to DIVA, a PIM-based data-intensive architecture,” in *Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, p. 57, 1999
`http://dl.acm.org/citation.cfm?id=331589`

[57] Ham, T. S., Lee, S. K., Keasling, J. D., & Arkin, A. P., “A tightly regulated inducible expression system utilizing the fim inversion recombination switch,” Biotechnology and Bioengineering, Vol. 94, p. 14, 2006
`http://onlinelibrary.wiley.com/doi/10.1002/bit.20916/abstract`

[58] Hennessy, J. L. & Patterson, D. A., *Computer architecture: a quantitative approach*, Elsevier, 2012

[59] Holler, M., Tam, S., Castro, H., & Benson, R., “An electrically trainable artificial neural network (ETANN) with 10240 ’floating gate’ synapses,” in *, International Joint Conference on Neural Networks, 1989. IJCNN*, pp. 191–196 vol.2, 1989

[60] Holler, M., Tam, S., Castro, H., & Benson, R., “An electrically trainable artificial neural network (etann) with 10240’floating gate’synapses,” in *Neural Networks, 1989. IJCNN., International Joint Conference on*, pp. 191–196, IEEE, 1989

[61] Holmes, D., Ripple, A., & Manheimer, M., “Energy-Efficient Superconducting Computing—Power Budgets and Requirements,” Applied Superconductivity, IEEE Transactions on, Vol. 23, 2013

[62] Indiveri, G., Chicca, E., & Douglas, R. J., “Artificial Cognitive Systems: From VLSI Networks of Spiking Neurons to Neuromorphic Cognition,” Cognitive Computation, Vol. 1, pp. 119–127, 2009
`http://link.springer.com/article/10.1007/s12559-008-9003-6`

[63] Indiveri, G., et al., “Neuromorphic Silicon Neuron Circuits,” Frontiers in Neuroscience, Vol. 5, 2011, PMID: 21747754 PMCID: PMC3130465
`http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3130465/`

[64] Jack Ng, Y. & Van Dam, H., “Limit to space-time measurement,” Modern Physics Letters A, Vol. 9, pp. 335–340, 1994

[65] JASON, “Technical Challenges of Exascale Computing,” JASON, Tech. Rep. JSR-12-310, 2013

[66] Jozsa, R., “Quantum factoring, discrete logarithms, and the hidden subgroup problem,” Computing in Science & Engineering, Vol. 3, pp. 34–43, 2001

[67] Khalil, A. S. & Collins, J. J., “Synthetic biology: applications come of age.” Nature reviews. Genetics, Vol. 11, pp. 367–79, 2010
http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2896386&tool=pmcentrez&rendertype=abstract

[68] Kitano, H., “Computational systems biology.” Nature, Vol. 420, pp. 206–10, 2002
http://www.ncbi.nlm.nih.gov/pubmed/17052114

[69] Kornberg, A. & Baker, e. a., Tania A, *DNA replication*, Vol. 5, WH Freeman New York, 1992

[70] Landauer, R., “Irreversibility and heat generation in the computing process,” IBM Journal of Research and Development, Vol. 5, pp. 183–191, 1961

[71] Lewis, N. & Weinberger, P., “DNA Computing,” JASON, Tech. rep., 1995

[72] Li, H., Xu, C., & Banerjee, K., “Carbon Nanomaterials: The Ideal Interconnect Technology for Next-Generation ICs,” IEEE Design Test of Computers, Vol. 27, pp. 20–31, 2010

[73] Li, X., Yang, X., Qi, J., & Seeman, N. C., “Antiparallel DNA Double Crossover Molecules As Components for Nanoconstruction,” Journal of the American Chemical Society, Vol. 118, pp. 6131–6140, 1996
http://pubs.acs.org/doi/abs/10.1021/ja960162o

[74] Likharev, K. K., “CrossNets: Neuromorphic Hybrid CMOS/Nanoelectronic Networks,” Science of Advanced Materials, Vol. 3, pp. 322–331, 2011

[75] Lima, S. Q. & Miesenböck, G., “Remote control of behavior through genetically targeted photostimulation of neurons,” Cell, Vol. 121, pp. 141–152, 2005

[76] Lipton, R. J., Boneh, D., & Dimworth, C., “Breaking DES Using a Molecular Computer,” Vol. 27, pp. 1–20, 1996

[77] Lloyd, S., “Ultimate physical limits to computation,” Nature, Vol. 406, pp. 1047–1054, 2000

[78] Lyon, R. & Mead, C., “An analog electronic cochlea,” IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 36, pp. 1119–1134, 1988

[79] MacLennan, B. J., “A review of analog computing,” Department of Electrical Engineering & Computer Science, University of Tennessee, Technical Report UT-CS-07-601 (September), 2007
ftp://ftp.cs.utk.edu/pub/maclennan/RAC-TR.pdf

[80] Mallik, U., Vogelstein, R., Culurciello, E., Etienne-Cummings, R., & Cauwenberghs, G., "A real-time spike-domain sensory information processing system [image processing applications]," in *IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005*, pp. 1919–1922 Vol. 3, 2005

[81] Margolus, N. & Levitin, L. B., "The maximum speed of dynamical evolution," Physica D: Nonlinear Phenomena, Vol. 120, pp. 188–195, 1998

[82] Marr, B., George, J., Degnan, B., Anderson, D. V., & Hasler, P., "Error Immune Logic for Low-power Probabilistic Computing," VLSI Des., Vol. 2010, pp. 6:1–6:9, 2010
`http://dx.doi.org/10.1155/2010/460312`

[83] Mead, C. A. & Mahowald, M. A., "A silicon model of early visual processing," Neural Networks, Vol. 1, pp. 91–97, 1988
`http://www.sciencedirect.com/science/article/pii/089360808890024X`

[84] Mills, J. W., et al., "Extended analog computers: A unifying paradigm for VLSI, plastic and colloidal computing systems," in *Workshop on Unique Chips and Systems (UCAS-1). Held in conjunction with IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS05), Austin, TX*, 2005

[85] Mitra, S., Fusi, S., & Indiveri, G., "Real-Time Classification of Complex Patterns Using Spike-Based Learning in Neuromorphic VLSI," IEEE Transactions on Biomedical Circuits and Systems, Vol. 3, pp. 32–42, 2009

[86] Miyamoto, T., Razavi, S., DeRose, R., & Inoue, T., "Synthesizing biomolecule-based Boolean logic gates." ACS synthetic biology, Vol. 2, pp. 72–82, 2013
`http://www.ncbi.nlm.nih.gov/pubmed/23526588`

[87] Miyamoto, T., Razavi, S., DeRose, R., & Inoue, T., "Synthesizing Biomolecule-Based Boolean Logic Gates," ACS Synthetic Biology, Vol. 2, pp. 72–82, 2013
`http://dx.doi.org/10.1021/sb3001112`

[88] Niazov, T., Baron, R., Katz, E., Lioubashevski, O., & Willner, I., "Concatenated logic gates using four coupled biocatalysts operating in series." Proceedings of the National Academy of Sciences of the United States of America, Vol. 103, pp. 17160–17163, 2006
`http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1634834&tool=pmcentrez&rendertype=abstract`

[89] Niemier, M. T., et al., "Nanomagnet logic: progress toward system-level integration," Journal of Physics: Condensed Matter, Vol. 23, p. 493202, 2011
`http://iopscience.iop.org/0953-8984/23/49/493202`

[90] Page, L., Brin, S., Motwani, R., & Winograd, T., “The PageRank citation ranking: Bringing order to the web,” , 1999

[91] Patterson, D., et al., “A case for intelligent RAM,” Micro, IEEE, Vol. 17, p. 3444, 1997
`http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=592312`

[92] Pfeil, T., et al., “Six networks on a universal neuromorphic computing substrate,” Frontiers in Neuroscience, Vol. 7, 2013, PMID: 23423583 PMCID: PMC3575075
`http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3575075/`

[93] Poon, C.-S. & Zhou, K., “Neuromorphic Silicon Neurons and Large-Scale Neural Networks: Challenges and Opportunities,” Frontiers in Neuroscience, Vol. 5, 2011, PMID: 21991244 PMCID: PMC3181466
`http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3181466/`

[94] Qian, L. & Winfree, E., “Scaling up digital circuit computation with DNA strand displacement cascades.” Science, Vol. 332, pp. 1196–201, 2011
`http://www.ncbi.nlm.nih.gov/pubmed/21636773`

[95] Qian, L., Winfree, E., & Bruck, J., “Neural network computation with DNA strand displacement cascades.” Nature, Vol. 475, pp. 368–72, 2011
`http://www.ncbi.nlm.nih.gov/pubmed/21776082`

[96] Rajendran, B., et al., “Specifications of nanoscale devices and circuits for neuromorphic computational systems,” IEEE Transactions on Electron Devices, Vol. 60, pp. 246–253, 2013

[97] Ramakrishnan, S., Hasler, P., & Gordon, C., “Floating Gate Synapses With Spike-Time-Dependent Plasticity,” IEEE Transactions on Biomedical Circuits and Systems, Vol. 5, pp. 244–252, 2011

[98] Reif, J. H., “Local parallel biomolecular computation,” DNA-Based Computers, Vol. 3, pp. 217–254, 1999

[99] Rinaudo, K., Bleris, L., Maddamsetti, R., Subramanian, S., Weiss, R., & Benenson, Y., “A universal RNAi-based logic evaluator that operates in mammalian cells,” Nature Biotechnology, Vol. 25, pp. 795–801, 2007
`http://www.nature.com/nbt/journal/v25/n7/abs/nbt1307.html`

[100] Robinson, R. M., “Undecidability and nonperiodicity for tilings of the plane,” Inventiones Mathematicae, Vol. 12, pp. 177–209, 1971
`http://link.springer.com/10.1007/BF01418780`

[101] Rothemund, P. W. K., “A DNA and restriction enzyme implementation of Turing machines,” pp. 75–120, 1996

[102] Rothemund, P. W. K., “Folding DNA to create nanoscale shapes and patterns.” Nature, Vol. 440, pp. 297–302, 2006
http://www.ncbi.nlm.nih.gov/pubmed/16541064

[103] Rothemund, P. W. K. & Winfree, E., “The Program-Size Complexity of Self-Assembled Squares,” pp. 1–10, 2000

[104] Rubel, L. A., “The brain as an analog computer,” Journal of theoretical neurobiology, Vol. 4, pp. 73–81, 1985

[105] Rubel, L. A., “The extended analog computer,” Advances in Applied Mathematics, Vol. 14, pp. 39–50, 1993

[106] Sarpeshkar, R., *Ultra low power bioelectronics*, Vol. 1, Cambridge University Press Cambridge, UK, 2010

[107] Schemmel, J., Brderle, D., Grbl, A., Hock, M., Meier, K., & Millner, S., “A wafer-scale neuromorphic hardware system for large-scale neural modeling,” in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1947–1950, 2010

[108] Schwierz, F., “Graphene transistors,” Nature Nanotechnology, Vol. 5, pp. 487–496, 2010
http://www.nature.com/nnano/journal/v5/n7/abs/nnano.2010.89.html

[109] Serrano-Gotarredona, R., et al., “CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory–processing–learning–actuating system for high-speed visual object recognition and tracking,” Neural Networks, IEEE Transactions on, Vol. 20, pp. 1417–1438, 2009

[110] Serrano-Gotarredona, R., et al., “CAVIAR: A 45k Neuron, 5M Synapse, 12G Connects/s AER Hardware Sensory Processing; Learning Actuating System for High-Speed Visual Object Recognition and Tracking,” IEEE Transactions on Neural Networks, Vol. 20, pp. 1417–1438, 2009

[111] Serrano-Gotarredona, T., Masquelier, T., Prodromakis, T., Indiveri, G., & Linares-Barranco, B., “STDP and STDP variations with memristors for spiking neuromorphic learning systems,” Frontiers in Neuroscience, Vol. 7, 2013, PMID: 23423540 PMCID: PMC3575074
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3575074/

[112] Shannon, C. E., “Mathematical theory of the differential analyzer,” J. Math. Phys. MIT, Vol. 20, pp. 337–354, 1941

[113] Sharad, M., Augustine, C., Panagopoulos, G., & Roy, K., “Proposal For Neuromorphic Hardware Using Spin Devices,” arXiv:1206.3227 [cond-mat], 2012
http://arxiv.org/abs/1206.3227

[114] Shi, J., Ha, S. D., Zhou, Y., Schoofs, F., & Ramanathan, S., "A correlated nickelate synaptic transistor," Nature Communications, Vol. 4, 2013 http://www.nature.com/ncomms/2013/131031/ncomms3676/full/ncomms3676.html

[115] Shor, P. W., "Why haven't more quantum algorithms been found?" Journal of the ACM (JACM), Vol. 50, p. 8790, 2003 http://dl.acm.org/citation.cfm?id=602408

[116] Siegelman, H. T., *Neural networks and analog computation: Beyond the Turing limit*, Vol. 20, Springer, 1999

[117] Smith, W. D., "DNA computers in vitro and vivo," Vol. 27, pp. 121–186, 1996

[118] Srivastava, N., Li, H., Kreupl, F., & Banerjee, K., "On the Applicability of Single-Walled Carbon Nanotubes as VLSI Interconnects," IEEE Transactions on Nanotechnology, Vol. 8, pp. 542–559, 2009

[119] Stepney, S., et al., "Journeys in non-classical computation I: A grand challenge for computing research," International Journal of Parallel, Emergent and Distributed Systems, Vol. 20, pp. 5–19, 2005, http://www.tandfonline.com/doi/pdf/10.1080/17445760500033291 http://www.tandfonline.com/doi/abs/10.1080/17445760500033291

[120] Sterling, T. L. & Zima, H. P., "Gilgamesh: a multithreaded processor-in-memory architecture for petaflops computing," in *Supercomputing, ACM/IEEE 2002 Conference*, p. 4848, 2002 http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1592884

[121] Stojanovic, M. N. & Stefanovic, D., "A deoxyribozyme-based molecular automaton." Nature Biotechnology, Vol. 21, pp. 1069–74, 2003 http://www.ncbi.nlm.nih.gov/pubmed/12923549

[122] Turing, A. M., "Computability and -Definability," The Journal of Symbolic Logic, Vol. 2, pp. pp. 153–163, 1937 http://www.jstor.org/stable/2268280

[123] Wang, H., "Proving theorems by pattern recognition I," Communications of the ACM, Vol. 3, pp. 220–234, 1960 http://portal.acm.org/citation.cfm?doid=367177.367224

[124] Weisstein, E. W., "Turing Machine – from Wolfram MathWorld," http://mathworld.wolfram.com/TuringMachine.html

[125] Win, M. N. & Smolke, C. D., "Higher-order cellular information processing with synthetic RNA devices." Science (New York, N.Y.), Vol. 322, pp. 456–60, 2008

http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2805114&tool=pmcentrez&rendertype=abstract

[126] Win, M. N. & Smolke, C. D., “Higher-Order Cellular Information Processing with Synthetic RNA Devices,” Science, Vol. 322, pp. 456–460, 2008, PMID: 18927397
http://www.sciencemag.org/content/322/5900/456

[127] Winfree, E., “Simulations of computing by self-assembly,” 1998

[128] Winfree, E., Liu, F., Wenzler, L. a., & Seeman, N. C., “Design and self-assembly of two-dimensional DNA crystals.” Nature, Vol. 394, pp. 539–44, 1998
http://www.ncbi.nlm.nih.gov/pubmed/9707114

[129] Xu, Y., et al., “Large-Gap Quantum Spin Hall Insulators in Tin Films,” Physical Review Letters, Vol. 111, p. 136804, 2013
http://link.aps.org/doi/10.1103/PhysRevLett.111.136804

[130] Zamarreno-Ramos, C., Linares-Barranco, A., Serrano-Gotarredona, T., & Linares-Barranco, B., “Multicasting Mesh AER: A Scalable Assembly Approach for Reconfigurable Neuromorphic Structured AER Systems. Application to ConvNets,” IEEE Transactions on Biomedical Circuits and Systems, Vol. 7, pp. 82–102, 2013

[131] Zhirnov, V., Cavin, R., Hutchby, J., & Bourianoff, G., “Limits to binary logic switch scaling - a gedanken model,” Proceedings of the IEEE, Vol. 91, pp. 1934–1939, 2003

[132] Zhou, Y. & Ramanathan, S., “Correlated Electron Materials and Field Effect Transistors for Logic: A Review,” Critical Reviews in Solid State and Materials Sciences, Vol. 38, pp. 286–317, 2013, arXiv:1212.2684 [cond-mat]
http://arxiv.org/abs/1212.2684

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE January 2014	2. REPORT TYPE Final	3. DATES COVERED *(From–To)* Apr 2013 – Jan 2014

4. TITLE AND SUBTITLE

An Initial Look at Alternative Computing Technologies
for the Intelligence Community

5a. CONTRACT NUMBER
HQ0034-14-D-0001

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Lance Joneckis, IDA
David Koester, MITRE Corporation
Joshua Alspector, IDA

5d. PROJECT NUMBER

5e. TASK NUMBER
ET-2-2954.22

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
4850 Mark Center Drive
Alexandria, VA 22311-1882

8. PERFORMING ORGANIZATION REPORT NUMBER

IDA Paper P-5114
Log: H 14-000128

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of the Director of National Intelligence
Intelligence Advanced Research Projects Activity
Washington, DC 20511

10. SPONSOR/MONITOR'S ACRONYM(S)

IARPA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited (20 May 2014).

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We have broadly surveyed the landscape of computing as it relates to the problems of interest to the intelligence community (IC). We find there is no single alternative computing technology (ACT) that is universally applicable to the wide range of IC applications, although there are particular solutions that are suited to individual application classes. We therefore recommend that the IC consider an application-driven, holistic design approach that spans a broad range of technologies and computational models. The application classes we considered were discrete math, big data, distributed sensing and processing, scientific/numerical simulation, robotics/autonomous systems. Holistic design could extend beyond a single technology to include integration of multiple, heterogeneous ACTs as needed to meet application/architecture requirements.

15. SUBJECT TERMS

alternative computing technology, analog computing, biomolecular computing, high-performance computing (HPC), neuro-inspired computing, neuromorphic computing, quantum computing, superconductive computing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Peter Highnam
a. REPORT Uncl.	b. ABSTRACT Uncl.	c. THIS PAGE Uncl.	SAR	86	19b. TELEPHONE NUMBER *(include area code)* (301) 851-7582